COSC 460
Honours Project
1991.

Written by: Lindsay Weir.

Supervisor: Ray Hunt.

Tellabs

Network

Monitor

-source code listing

Mount Cook Group Ltd.

TELLABS NETWORK MONITOR

SOURCE CODE LISTING

1. INSTALLATION

- README
- INSTALL.BAT

2. INTERRUPT HANDLER ROUTINES

- COMINT.ASM
- COMIO.C

3. EVENT LOG PROCESS

- EVENTLOG.H
- EVENTLOG.LEX
- EVENTLOG.YAC
- EVENTCFG.DAT
- EXAMPLE1.TXT
- LEXYY.C
- Y_TAB.C

4. COMMAND PORT PROCESS

- CMDPORT.LEX
- CMDPORT.YAC
- CMDPORT.C
- COMMDCFG.DAT
- DATA005.BAT

  .
  .
  .
- DATA203.BAT
- EXAMPLE2.TXT
- LEXYY.C
- Y_TAB.C

5. DATABASE CONSTRUCTION

5.1 DATABASE FILE STRUCTURES

5.2 DATABASE QUERY STRUCTURES

5.3 TELLABS APPLICATION PROGRAM

- (Example showing menu layouts, forms, and reports) see TELLABS NETWORK MONITOR - USERS GUIDE.
- TXTTOTMP.C
- DELFILES.C

Note: Sections may not have headings but are colour divided.

TELLABS NETWORK MONITOR

To install the Tellabs Network Monitor simply run the INSTALL.BAT batch file. This reads (by default) from the original source disk in drive B: and writes to the destination hard drive on drive C:. If any changes are needed then simply edit the install.bat file with a text editor like EDLIN or MICRO-EMACS.

Contents of the Directories:

\TELLABS\COMPORTS

EVENTLOG.EXE	-	eventlog process
CMDPORT.EXE	-	command port process
EVENTCFG.DAT	-	eventlog configuration data file
COMMDCFG.DAT	-	command port configuration data file
EXAMPLE1.DAT	-	template of how the two configuration files should be set up
DATA003.BAT DATA004.BAT DATA005.BAT DATA100.BAT DATA101.BAT DATA102.BAT DATA107.BAT DATA108.BAT DATA109.BAT DATA202.BAT DATA203.BAT	-	default dataplexer interrogation batch files. (These contain the key commands which would be entered from a command terminal to TELLABS). These are valid as of 01/10/91.
EXAMPLE2.DAT	-	template of what is in the configuration and batch files and how to and how to create new batch files.

\TELLABS\DATABASE

TELLABS.DBO	-	executable dBase IV program
TXTTOTMP.EXE	-	program used by TELLABS.DBO to copy the text database files produced by EVENTLOG.EXE and CMDPORT.EXE in \TELLABS\TXTFILES to a temporary file. From here updates to the database can be done at leisure. After copying the contents of *.TXT to *.TMP the file size of *.TXT is set to zero. This is so any new data is written to a new file while the existing *.TMP file can be updated and the *.TMP file can be deleted.
DELFILES.TXT	-	program used by the TELLABS.DBO to delete files which contain captured data which is not being used by the database.

RUNTIME.* - runtime library to run the executable
TELLABS.DBO file.

Other associated database files.

\WINDOWS

EVENTLOG.PIF - PIF files for setting up multi-tasking
CMDPORT.PIF - options for windows.
TELLABS.PIF

```
@echo off
echo Installing Tellabs Network Monitor
md C:\TELLABS
md C:\TELLABS\COMPORTS
md C:\TELLABS\BATFILES
md C:\TELLABS\DATABASE
md C:\TELLABS\TXTFILES
copy B:\TELLABS\COMPORTS\*.* C:\TELLABS\COMPORTS
copy B:\TELLABS\BATFILES\*.* C:\TELLABS\BATFILES
copy B:\TELLABS\DATABASE\*.* C:\TELLABS\DATABASE
echo Installation of Tellabs Network Monitor Complete
@echo on
```

```
page 41,128
title Interrupt routines for Micrisoft mouse.

;*****************************************************************************

;       FILENAME    : comint.asm
;       DATE        : 88-10-28.
;       AUTHOR      : Hans Nasten, Everyware Mikrodata AB.
;
;       MODIFIED BY         Lindsay Weir, 181 West St, Invercargill.
;
;       DESCRIPTION : Interrupt driver used with com ports.

;*****************************************************************************


;-----------------------------------------------------------------------------

;       Constants et.c.

;-----------------------------------------------------------------------------

MSDOS           equ     21h             ; Dos function call interrupt.
COM1INT         equ     12              ; COM1 interrupt number.
GET_VECTOR      equ     35h             ; Get interrupt vector funct. no.
SET_VECTOR      equ     25h             ; Set interrupt vector funct. no.
EOI             equ     20h             ; End of interrupt command.
I8259           equ     20h             ; 8259 interrupt controller.
Max_Buffer      equ     7FFFh           ; Maximum buffer size: 0 - 32767.

;       Offsets in 8250 chip.


IIR             equ     2               ; Address to int. id register in 8250.
LCR             equ     3               ; Address to line control register.
MCR             equ     4               ; Address to modem control register.
LSR             equ     5               ; Address to line status register.
MSR             equ     6               ; Address to modem status register.


;       Ascii control characters.


ACK             equ     06h             ; Ack block id character.
NAK             equ     15h             ; Nak block id character.
STX             equ     02h             ; Data block id character.
ENQ             equ     05h             ; Poll block id character.

;*****************************************************************************

;       References in Microsoft C:s data segment.

;*****************************************************************************

_DATA segment word public 'DATA'

extrn   _input_fifo  : byte
extrn   _fifo_size   : word
extrn   _fifo_index  : word
extrn   _xmt_ready   : word
```

```
extrn   _output_pnt  : dword
extrn   _output_size : byte
extrn   _com_port    : word

_DATA ends

DGROUP group _DATA



;*************************************************************************

;       Start of code segment.

;*************************************************************************


COM_TEXT  segment byte public 'CODE'
              assume cs:COM_TEXT, ds:DGROUP, es:nothing, ss:nothing

public _irq_entry


;-------------------------------------------------------------------------

;       Jump table to reach all 4 types of com interrupts.

;-------------------------------------------------------------------------


entry_points  dw      modem_status
              dw      tx_empty
              dw      rx_full
              dw      rx_status


;*************************************************************************

;       Entrypoint to com port interrupt routine.

;*************************************************************************


_irq_entry proc far

        push    ax                      ; Save all registers used in
        push    bx                      ; interrupt routine.
        push    dx
        push    ds
        mov     ax,DGROUP               ; Point at C:s data segment.
        mov     ds,ax

irq_exit:
        mov     dx,_com_port            ; Get the base address to the 8250.
        add     dx,IIR                  ; Point at the interrupt ident. reg.
        xor     ax,ax
```

```
        in      al,dx                   ; Read the interrupt id byte.
        test    al,00000001b            ; Pending interrupts ?
        jnz     irq_terminate           ; No, skip this interrupt.

        mov     bx,ax                   ; Yes, get address to routine.
        jmp     entry_points[bx]        ; Jump to proper interrupt routine.

;-------------------------------------------------------------------------

;       Common exit point.

;-------------------------------------------------------------------------

irq_terminate:
        mov     al,EOI                  ; Send "end of interrupt" to 8259.
        out     I8259,al
        pop     ds                      ; Restore saved registers.
        pop     dx
        pop     bx
        pop     ax
        iret


;*************************************************************************

;       Receive character available routine.

;*************************************************************************

rx_full:
        mov     dx,_com_port                    ; Read port address.
        in      al,dx                           ; Get received character.
        mov     bx,_fifo_index                          ; Get buffer index.
        mov     _input_fifo[bx],al              ; Save character in buffer.
        cmp     _fifo_index, Max_Buffer                 ; End of Buffer.
        jne     increm
        mov     _fifo_index, 0                          ; Reset index into the buffer.
        inc     _fifo_size                      ; Increment buffer size.
        jmp irq_exit

increm:
        inc     _fifo_index                     ; Advance fifo pointer
        inc     _fifo_size                      ; Increment buffer index.
        jmp     irq_exit


;*************************************************************************

;       Transmit holding register empty routine.

;*************************************************************************

tx_empty:
        cmp     _output_size,0                          ; All characters sent ?
        jnz     tx_send_next                    ; No, send next character.

tx_all_sent:
        mov     _xmt_ready,1                    ; Set "complete" status.
```

```
        jmp     irq_exit

tx_send_next:
        push    es
        les     bx,_output_pnt                  ; Get buffer pointer.
        mov     al,es:[bx]                      ; Get character from buffer.
        mov     dx,_com_port
        out     dx,al                           ; Send character to port.
        inc     bx
        mov     word ptr _output_pnt,bx         ; Increment buffer pointer.
        dec     _output_size                    ; Decrement character counter.
        pop     es
        jmp     irq_exit


;*****************************************************************************

;       Mode status interrupt routine. (not yet used).

;*****************************************************************************

modem_status:
        mov     dx,_com_port
        add     dx,MSR                          ; Point at modem status register.
        in      al,dx                   ; Clear interrupt.
        jmp     irq_exit


;*****************************************************************************

;       Line status interrupt routine. (not yet used).

;*****************************************************************************

rx_status:
        mov     dx,_com_port
        add     dx,LSR                          ; Point at line status register.
        in      al,dx                   ; Clear interrupt.
        jmp     irq_exit


_irq_entry endp


COM_TEXT ends

        end
```

```
/*****************************************************************************

        FILE        : comio.c
        DATE        : 88-10-28.
        AUTHOR      : Hans Nasten, Everyware Mikrodata AB.

        MODIFIED BY         Lindsay Weir, 181 West St, Invercargill.

        DESCRIPTION : Input / ouput routines for com ports.

        ENTRYPOINTS : open_com          - initialize.
                      close_com         - restore previous conditions.
                      test_com_input    - check to see if input pending.
                      read_com          - wait for chars from port.
                      com_output_ready  - check if output is ready.
                      start_com_output  - start backgr. output.

*****************************************************************************/


#include <d:\qc2\include\stddef.h>
#include <d:\qc2\include\dos.h>
#include <d:\qc2\include\malloc.h>


#define I8259 0x21              /* Address of 8259 interrupt controller.    */
#define COM1_ON 0xef                    /* Mask to turn INT 4 on. ( COM1 ).         */
#define COM1_OFF 0x10                   /* Mask to turn INT 4 off. ( COM1 ).        */
#define COM1_INT 12                     /* COM1 interrupt number.               */
#define COM1_ADDR 0x3f8                         /* COM1 hardware address.           */
#define COM2_ON 0xf7                    /* Mask to turn INT 3 on. ( COM2 ).         */
#define COM2_OFF 0x08                   /* Mask to turn INT 3 off. ( COM2 ).        */
#define COM2_INT 11                     /* COM2 interrupt number.               */
#define COM2_ADDR 0x2f8                         /* COM2 hardware address.           */
#define Max_Buffer_Size 32767   /* Maximum buffer size for receiving data   */
#define FALSE 0
#define TRUE 1



/*---------------------------------------------------------------------------

        External functions.
        These functions is found in the interrupt driver module.

---------------------------------------------------------------------------*/


extern void far interrupt irq_entry();          /* Interrupt entrypoint.            */



/*---------------------------------------------------------------------------

        Internal static data declarations.

---------------------------------------------------------------------------*/


                                /* Init sequence for all channels.          */
```

```
static char com_init[][2] = { { 3, 0x83 },      /* Set DLAB.              */
                              { 1, 0x00 },      /* Set baudrate MSB.      */
                              { 0, 0x60 },      /* Set baudrate LSB.      */
                              { 3, 0x03 },      /* Reset DLAB.                  */
                              { 4, 0x0b },      /* Int driver & DTR/RTS on. */
                              { 1, 0x03 }       /* TX & RX int's on.      */
};

                              /* Reset sequence for all channels.      */


static char com_res[][2] =  { { 4, 0x00 },      /* Int driver & DTR/RTS off.*/
                              { 1, 0x00 }       /* All int's off.         */
};

                              /* Hardware port addresses.              */

static unsigned int com_addresses[] = { COM1_ADDR,      /* COM1 address.    */
                                        COM2_ADDR       /* COM2 address.    */
};

                              /* Hardware interrupt numbers.                  */

static unsigned int com_interrupts[] = { COM1_INT,      /* COM1 int.number. */
                                         COM2_INT  /* COM2 int.number. */
};

                              /* Hardware interrupt masks.             */

static unsigned int com_on_masks[] = { COM1_ON,             /* COM1 int.active. */
                                       COM2_ON              /* COM2 int.active. */
};

static unsigned int com_off_masks[] = { COM1_OFF,       /* COM1 int.passive.*/
                                        COM2_OFF    /* COM2 int.passive.*/
};


                     /* Speed constants used to set the baudrate.      */


static unsigned int speed_table[] = { 0x180,        /* 300 baud.      */
                                      0x0c0,        /* 600 baud.      */
                                      0x060,        /* 1200 baud.     */
                                      0x030,        /* 2400 baud.     */
                                      0x018,        /* 4800 baud.     */
                                      0x00c,        /* 9600 baud.     */
                                      0x006,        /* 19200 baud.          */
                                      0x003         /* 38400 baud.          */
};

                     /* Word length constants for UART initialization.   */


static unsigned char word_length_tab[] = { 0x02,    /* 7 bits, no parity*/
                                           0x03     /* 8 bits, no parity*/
};
```

```
static int near port_no;
unsigned int near read_index = 0;
static void (interrupt far *old_vector)( void );

unsigned char near input_fifo[ Max_Buffer_Size ]; /* Input fifo from interrupt driver.*/
unsigned int near fifo_size = 0;       /* Number of characters in inp.fifo.*/
unsigned int near fifo_index = 0;      /* Index in input fifo.             */
int near xmt_ready = TRUE;         /* Transmission ready flag.       */
unsigned char far * near output_pnt;        /* Output buffer pointer.         */
unsigned char near output_size = 0; /* Size of data in output buffer.   */
unsigned int near com_port = 0x3f8;/* COM port address.               */


/*************************************************************************

       FUNCTION    : test_com_input.

       ARGUMENTS   : none.

       RETURNS     : number of characters in input fifo.

       DESCRIPTION : Check input status.

*************************************************************************/

int test_com_input()

       {

         return( (unsigned int) ( fifo_size ) );

       }



/*************************************************************************

       FUNCTION    : read_com.

       ARGUMENTS   : none.

       RETURNS     : a character from the com port.

       DESCRIPTION : wait for a character from the com port.

*************************************************************************/

unsigned char read_com()

       {
       unsigned char ch;

         while( test_com_input() == 0 )
           ;

         _disable();
         ch = input_fifo[ read_index ];
         if (read_index == Max_Buffer_Size)
```

```
            read_index = 0;
        else
            read_index = read_index + 1;
        fifo_size--;
        _enable();
        return( ch );

    }



/*****************************************************************************

    FUNCTION    : com_output_ready.

    ARGUMENTS   : none.

    RETURNS     : state of output operation.

    DESCRIPTION : This entrypoint returns the present output status.

*****************************************************************************/

int com_output_ready()

    {

      return( xmt_ready );

    }



/*****************************************************************************

    FUNCTION    : start_com_output.

    ARGUMENTS   : length      - number of characters in buffer.

    RETURNS     : TRUE / FALSE.

                  TRUE                  = transmission started OK.
                  FALSE           = transmission already started.

    DESCRIPTION : This entrypoint starts a output operation.

*****************************************************************************/

int start_com_output( pnt, length )
char *pnt;
int length;


    {
    char ch;
```

```
    if ( !com_output_ready() )
      return( FALSE );
    else {
      _disable();
      ch = *pnt++;
      output_pnt = pnt;
      output_size = (unsigned char)(length - 1);
      xmt_ready = FALSE;
      outp( com_port, ch );
      _enable();
      return( TRUE );
    }


  }


/*****************************************************************************

    FUNCTION    : write_to_com.

    ARGUMENTS   : ch - character to write.

    RETURNS     : nothing.

    DESCRIPTION : Send a byte to the com port.

*****************************************************************************/

void write_to_com( ch )
char ch;


  {

    if( start_com_output( &ch, 1 ) )
      while( !com_output_ready() )
        ;

  }



/*****************************************************************************

    FUNCTION    : open_com.

    ARGUMENTS   : speed         - an integer value to indicate baudrate
                                  for the com ports.
                                  0 = 300 baud.
                                  1 = 600 baud.
                                  2 = 1200 baud.
                                  3 = 2400 baud.
                                  4 = 4800 baud.
                                  5 = 9600 baud.
                                  6 = 19200 baud.
                                  7 = 38400 baud.
```

```
                port_number     - 1 = COM1.
                                2 = COM2.

                word_length     - 7 = 7 bits, ingen paritet.
                                8 = 8 bits, ingen paritet.

        RETURNS     : TRUE/FALSE.

        DESCRIPTION : This entrypoint initializes and starts the
                i/o-driver.

*************************************************************************/

int open_com( speed, port_number, word_length )
unsigned int speed, port_number, word_length;


        {
        int i;

          if( speed > (sizeof speed_table / 2) - 1 )  /* Check speed arg. */
            return( FALSE );
          else {                                    /* Set speed.       */
            com_init[ 1 ][ 1 ] = speed_table[ speed ] >> 8;
            com_init[ 2 ][ 1 ] = speed_table[ speed ] & 0x00ff;
          }

          if( word_length != 7 && word_length != 8 )      /* Check word length*/
            return( FALSE );
          else {                                    /* Set word length. */
            com_init[ 0 ][ 1 ] = word_length_tab[ word_length - 7 ] | 0x80;
            com_init[ 3 ][ 1 ] = word_length_tab[ word_length - 7 ];
          }

          if( ( port_number != 1 ) && ( port_number != 2 ) )
            return( FALSE );

          port_no = port_number - 1;
          xmt_ready = TRUE;
          com_port = com_addresses[ port_no ];
          fifo_index = fifo_size = read_index = 0;


                        /*-------------------------------------------------*/
                        /* Then initialize the com port hardware.       */
                        /* The init values to control port speed has been  */
                        /* set using the value from in argument "speed".   */
                        /*-------------------------------------------------*/

          _disable();
          for( i = 0; i < sizeof com_init / 2; i++ )
            outp( com_port + com_init[ i ][ 0 ], com_init[ i ][ 1 ] );

          inp( com_port );              /* Clear receive register   */
          inp( com_port );

          old_vector = _dos_getvect( com_interrupts[ port_no ] );
          _dos_setvect( com_interrupts[ port_no ], irq_entry );

                                        /* Reset mask bit in 8259.  */
```

```
        outp( I8259, inp( I8259 ) & com_on_masks[ port_no ] );

        _enable();
        return( TRUE );

      }


/*************************************************************************

      FUNCTION    : close_com.

      ARGUMENTS   : none.

      RETURNS     : nothing.

      DESCRIPTION : This function is called upon to restore the changes
                    to the MSDOS environment etc. made by the i/o-driver.

*************************************************************************/

void close_com()


      {
      int ch, i;

                          /* Restore hardware and interrupt vector.   */

        _disable();
                                                  /* Mask int in 8259.*/
        outp( I8259, inp( I8259 ) | com_off_masks[ port_no ] );
        for( i = 0; i < sizeof com_res / 2; i++ )
          outp( com_port + com_res[ i ][ 0 ],
                com_res[ i ][ 1 ] );

        _dos_setvect( com_interrupts[ port_no ], old_vector );

        _enable();


      }
```

```
# define DATE_TIME_LENGTH 12
# define XPLEX02_LENGTH 22
# define XPLEX03_LENGTH 25
# define XPLEX04_LENGTH 42
# define XPLEX05_LENGTH 28
# define XPLEX06_LENGTH 48
# define XPLEX07_LENGTH 30
# define XPLEX08_LENGTH 38
# define XPLEX10_LENGTH 106
# define XPLEX11_LENGTH 109
# define XPLEX12_LENGTH 37
# define XPLEX13_LENGTH 42
# define TRUE 1
# define FALSE 0
# define S_IWRITE    0000200        /* write permission, owner */

char dte[9];
char tme[5];
char time_stmp[13];
char previous_time[5];

char code[3];
char code_event[5];

int dev_type;
int event_code;

char node[4];
char link[4];
char channel[4];

char orig_node[4];
char orig_link[4];
char orig_channel[4];

char dl_string[21];

int queue_place;

unsigned int numb;

int count;

int speed;
int port_number;
int word_length;

char path_name[80];

int handle;
long int length;

int stage10;
int stage11;

int string_length;

char buffer_temp[30];
char file_buffer[100];
char *buffer;
```

```
char *buffer10;
char *buffer11;

struct input_stack {
                    char inputChar[256];
                    int top;
                   } input_stackptr;
```

```
%{

char pop()
{
        if (input_stackptr.top == 0)
          printf("Trying to pop a character off the stack which doesn't exist\n");
        else
          return(input_stackptr.inputChar[-- input_stackptr.top]);
}

push(c)
        char c;
{
        if (input_stackptr.top == 256)
          printf("Stack overflow in unput()\n");
        else
          input_stackptr.inputChar[input_stackptr.top ++] = c;
}

int empty()
{
        if (input_stackptr.top == 0)
          return(TRUE);
        else
          return(FALSE);
}

# undef input
input()
{
```

```
char return_value;
        if (empty())
          {
            return_value = read_com();
            return(return_value);
          }
        else
        {
          return_value = pop();
          return(return_value);
        }
}

# undef unput
unput(c)
char c;
{
        push(c);
}

%}

WS      [ \t]
D       [0-9]

%p 3000
%n 600

%%

\r
```

```
\n                      { printf("%s", yytext); return(NEW_LINE); }
{WS}+                   { printf("%s", yytext); }
"Calls"                 { printf("%s", yytext); }
"LC Buffer Util."           { printf("%s", yytext); return(LC_COMMENT); }
"Tx"                    { printf("%s", yytext); return(TRANS_COMMENT); }
"Rx"                    { printf("%s", yytext); return(COMMENT); }
"Channel"               { printf("%s", yytext); return(COMMENT); }
"Count"                     { printf("%s", yytext); return(COMMENT); }
"Call"                  { printf("%s", yytext); return(COMMENT); }
"Call Connect"          { printf("%s", yytext); return(COMMENT); }
"Xplexer"               { printf("%s", yytext); return(COMMENT); }
"New"                   { printf("%s", yytext); return(COMMENT); }
"Time U"                { printf("%s", yytext); return(COMMENT); }
"Database"                  { printf("%s", yytext); return(COMMENT); }
"Link"                      { printf("%s", yytext); return(COMMENT); }
"Buffer(s)"             { printf("%s", yytext); return(COMMENT); }
"Sys."                  { printf("%s", yytext); return(COMMENT); }
" m "                   { printf("%s", yytext); return(COMMENT); }
" A "                   { printf("%s", yytext); return(AVE_COMMENT); }
" P "                   { printf("%s", yytext); return(COMMENT); }
" T "                   { printf("%s", yytext); return(TRANS_COMMENT); }
"Characters(x1000)"         { printf("%s", yytext); }
"I Frames(x100)"        { printf("%s", yytext); }
"S Frames(x100)"            { printf("%s", yytext); }
"E Frames"                  { printf("%s", yytext); }
"Tx Utilization %"          { printf("%s", yytext); }
"Rx Utilization %"          { printf("%s", yytext); }
"Data"                  { printf("%s", yytext); return(COMMENT); }
"Data Link"             { printf("%s", yytext); return(COMMENT); }
"Impossible"            { printf("%s", yytext); return(COMMENT); }
```

```
"Network"                   { printf("%s", yytext); return(COMMENT); }
"Illegal"                       { printf("%s", yytext); return(COMMENT); }
"Number"                    { printf("%s", yytext); return(COMMENT); }
"Name Not Found"            { printf("%s", yytext); return(COMMENT); }
"Destination"               { printf("%s", yytext); return(COMMENT); }
"Not"                       { printf("%s", yytext); return(COMMENT); }
"Invalid"                   { printf("%s", yytext); return(COMMENT); }
"Network"                   { printf("%s", yytext); return(COMMENT); }
"Hop"                       { printf("%s", yytext); return(COMMENT); }
"Administration"                { printf("%s", yytext); return(COMMENT); }
"Operator"                  { printf("%s", yytext); return(COMMENT); }
"Unknown"                   { printf("%s", yytext); return(COMMENT); }
"Incomplete Datagram"           { printf("%s", yytext); return(COMMENT); }
"Datagram Checksum Error"   { printf("%s", yytext); return(COMMENT); }
"(Normal Disconnect)"           { printf("%s", yytext); return(BLANK_COMMENT); }
"(Data Link Down)"              { printf("%s", yytext); return(BLANK_COMMENT); }
"(Network Failure)"             { printf("%s", yytext); return(BLANK_COMMENT); }
"(Administration Override)"     { printf("%s", yytext); return(BLANK_COMMENT); }
"(Call Timed Out)"              { printf("%s", yytext); return(BLANK_COMMENT); }
"(Unknown Disconnect Code)"     { printf("%s", yytext); return(BLANK_COMMENT); }
"STANDBY"                   ;

"("{D}{D}"/"{D}{D}"/"{D}{D}{WS}{D}{D}":"{D}{D}")"   ;

{D}{D}"/"{D}{D}"/"{D}{D}    { printf("%s", yytext);
                              *(yytext + 2) = '\0';
                              *(yytext + 5) = '\0';
                              if (atoi(yytext + 6) >= 80)
                                  strcpy(dte, "19");
                              else
                                  strcpy(dte, "20");
```

```
                        strcat(dte, (yytext + 6));
                        strncat(dte, (yytext), 2);
                        strncat(dte, (yytext + 3), 2);
                        return(DATE); }

{D}{D}":"{D}{D}         { printf("%s", yytext);
                        *(yytext + 2) = '\0';
                        strcpy(tme, yytext);
                        strncat(tme, (yytext + 3), 2);
                        return(TIME); }

331|440|445     { printf("%s", yytext);
                 yylval = atoi(yytext);
                  return(DEVICE_TYPE); }

":"{D}{D}"."{D}{D}      { printf("%s", yytext);
                        *(yytext + 3) = '\0';
                        strcpy(code_event, (yytext + 1));
                        strncat(code_event, (yytext + 4), 2);
                        switch (atoi(code_event)) {
                                case 100:       yylval = 1;
                                                break;
                                case 200:       yylval = 2;
                                                break;
                                case 300:       yylval = 3;
                                                break;
                                case 402:       yylval = 4;
                                                break;
                                case 403:       yylval = 5;
                                                break;
```

```
case 404:       yylval = 6;
                break;
case 405:       yylval = 7;
                break;
case 406:       yylval = 8;
                break;
case 407:       yylval = 9;
                break;
case 500:       yylval = 10;
                break;
case 600:       yylval = 11;
                break;
case 700:       yylval = 12;
                break;
case 800:       yylval = 13;
                break;
case 900:       yylval = 14;
                break;
case 1000:      yylval = 15;
                break;
case 1100:      yylval = 16;
                break;
case 1200:      yylval = 17;
                break;
case 1300:      yylval = 18;
                break;
case 1401:      yylval = 19;
                break;
case 1402:      yylval = 20;
                break;
```

```
case 1403:      yylval = 21;
                break;
case 1404:      yylval = 22;
                break;
case 1405:      yylval = 23;
                break;
case 1406:      yylval = 24;
                break;
case 1407:      yylval = 25;
                break;
case 1408:      yylval = 26;
                break;
case 1409:      yylval = 27;
                break;
case 1410:      yylval = 28;
                break;
case 1411:      yylval = 29;
                break;
case 1500:      yylval = 30;
                break;
case 1501:      yylval = 31;
                break;
case 1510:      yylval = 32;
                break;
case 1511:      yylval = 33;
                break;
case 1600:      yylval = 34;
                break;
case 1700:      yylval = 35;
                break;
```

```
case 1800:      yylval = 36;
                break;
case 1801:      yylval = 37;
                break;
case 1812:      yylval = 38;
                break;
case 1814:      yylval = 39;
                break;
case 1817:      yylval = 40;
                break;
case 1819:      yylval = 41;
                break;
case 1901:      yylval = 42;
                break;
case 1902:      yylval = 43;
                break;
case 1903:      yylval = 44;
                break;
case 1904:      yylval = 45;
                break;
case 1905:      yylval = 46;
                break;
case 1906:      yylval = 47;
                break;
case 1907:      yylval = 48;
                break;
case 1908:      yylval = 49;
                break;
case 1909:      yylval = 50;
                break;
```

```
                case 1910:      yylval = 51;
                                break;
                case 1911:      yylval = 52;
                                break;
                case 1912:      yylval = 53;
                                break;
                case 1913:      yylval = 54;
                                break;
                case 1914:      yylval = 55;
                                break;
                case 1915:      yylval = 56;
                                break;
                case 1916:      yylval = 57;
                                break;
                case 1917:      yylval = 58;
                                break;
                case 1918:      yylval = 59;
                                break;
                case 1919:      yylval = 60;
                                break;
                case 2000:      yylval = 61;
                                break;
                case 2100:      yylval = 62;
                                break;
                case 2200:      yylval = 63;
                                break;
        }
        return(CODE); }

{D}{D}{D}       { printf("%s", yytext);
```

```
                yylval = atoi(yytext);
                return(ORIGINATING_NODE_NO); }


{D}{D}{D}"/"{D}{D}{D}."{D}{D}{D}{WS}+{D}{D}{D}"/CONTROL"     {
                                        printf("%s", yytext);
                                        *(yytext + 3) = '\0';
                                        *(yytext + 7) = '\0';
                                        *(yytext + 11) = '\0';
                                        *(yytext + 15) = '\0';
                                        strcpy(orig_node, yytext);
                                    strcpy(orig_link, (yytext + 4));
                                strcpy(orig_channel, (yytext + 8));
                                        strcpy(node, yytext + 12);
                                        strcpy(link, "CON");   /* CONTROL */
                                        strcpy(channel, "---");/* blank */
                                    return(CONTROL_ADDRESS); }


{D}{D}{D}"/."{D}{D}{D}        { printf("%s", yytext);
                        *(yytext + 3) = '\0';
                        trcpy(node, yytext);
                    strcpy(link, "---");       /* blank */
                        strcpy(channel, (yytext + 5)); /* X.25 or other */
                        return(X_25_LINK); }


{D}{D}{D}"/"{D}{D}{D}   { printf("%s", yytext);
                        *(yytext + 3) = '\0';
                        strcpy(node, yytext);
                    strcpy(link, (yytext + 4));
```

```
                return(LINK_ADDRESS); }

{D}{D}{D}"/"{D}{D}{D}"."{D}{D}{D}       { printf("%s", yytext);
                                          *(yytext + 3) = '\0';
                                         *(yytext + 7) = '\0';
                                         strcpy(node, yytext);
                                         strcpy(link, (yytext + 4));
                                         strcpy(channel, (yytext + 8));
                                          return(ADDRESS); }

{D}{D}{D}"/"{D}{D}{D}"."{D}{D}{D}{WS}+{D}{D}{D}"/"{D}{D}{D}"."{D}{D}{D}       {
                                          printf("%s", yytext);
                                          *(yytext + 3) = '\0';
                                         *(yytext + 7) = '\0';
                                          *(yytext + 11) = '\0';
                                          *(yytext + 15) = '\0';
                                          *(yytext + 19) = '\0';
                                         strcpy(orig_node, yytext);
                                         strcpy(orig_link, (yytext + 4));
                                         strcpy(orig_channel, (yytext + 8));
                                          strcpy(node, (yytext + 12));
                                         strcpy(link, (yytext + 16));
                                         strcpy(channel, (yytext + 20));
                                          return(CALL_ADDRESS); }

\"[^"]*\"       { printf("%s", yytext);
                  *(yytext + (yyleng - 1)) = '\0';
                  strcpy(dl_string, (yytext + 1));
                  return(DIAL_STRING); }
```

```
[1-9][0-9]*|[0]          { printf("%s", yytext);
                           numb = atoi(yytext);
                         yylval = atoi(yytext);
                           return(NUMB); }
"*****"                  { printf("%s", yytext); numb = 65536; return(NUMB); }
%%
```

```
%{
# include <h:\users\lbw\c600\include\stdio.h>
# include <h:\users\lbw\c600\include\string.h>
# include <h:\users\lbw\c600\include\io.h>
# include <h:\users\lbw\c600\include\sys\locking.h>
# include <h:\users\lbw\c600\include\fcntl.h>
# include <h:\users\lbw\c600\include\process.h>
# include <h:\users\lbw\c600\include\share.h>
# include <h:\users\lbw\c600\include\graph.h>
# include <h:\users\lbw\c600\include\malloc.h>
# include <h:\users\lbw\c600\include\dos.h>
# include <h:\users\lbw\c600\include\errno.h>
# include "event.h"

FILE *fp;
%}

%start event_log

%token DEVICE_TYPE MAJR_CODE MINOR_CODE ORIGINATING_NODE_NO ADDRESS
COMMENT
%token OPTIONAL_EXTENSION DESCRIPTOR LINK_ADDRESS CODE DATE TIME NUMB
NEW_LINE
%token DIAL_STRING CONTROL_ADDRESS START_UP_MESSAGE AVE_COMMENT
TRANS_COMMENT
%token CALL_ADDRESS BLANK_COMMENT LC_COMMENT

%%

event_log : time_based_datagram.
          | event_log time_based_datagram
          | error NEW_LINE { yyerrok; } event_log
          ;

time_stamp : DATE TIME NEW_LINE

           {
           strcpy(time_stmp, dte);
           strcat(time_stmp, tme);
           while ((handle = sopen("DATETIME.TXT", O_CREAT | O_WRONLY, SH_DENYNO,
S_IWRITE)) == FALSE);
           while (locking(handle, LK_NBRLCK, (long) DATE_TIME_LENGTH) != -1);
           lseek(handle, 0L, SEEK_SET);
           write(handle, time_stmp, DATE_TIME_LENGTH);
           lseek(handle, 0L, SEEK_SET);
           while (locking(handle, LK_UNLCK, (long) DATE_TIME_LENGTH) != -1);
           close(handle);
           }
           ;

time_based_datagram :      time_stamp datagram_list
                    ;

datagram_list : xplexer_datagram
              | datagram_list xplexer_datagram
              ;

xplexer_datagram : BLANK_COMMENT NEW_LINE
                 | device_type CODE ORIGINATING_NODE_NO minutes_ average_ peak_
last_
```

```
                {
                if (strcmp(tme, previous_time) == 0)
                  count = count + 1;
                else { strcpy(previous_time, tme);
                  count = 1; }
                switch (stage10) {
                  case 1: if ($2 == 19)
                            { sprintf(buffer10, "%s%3d%d%s%5u%2d%3d%3d%3d", time_stmp,
count, $1, node, $4, $2, $5, $6, $7);
                              stage10++; }
                              break;
                  case 2: if ($2 == 20)
                            { sprintf(buffer_temp, "%2d%3d%3d%3d", $2, $5, $6, $7);
                              strcat(buffer10, buffer_temp);
                              stage10++; }
                          else
                            stage10 = 1;
                            break;
                  case 3: if ($2 == 21)
                            { sprintf(buffer_temp, "%2d%3d%3d%3d", $2, $5, $6, $7);
                              strcat(buffer10, buffer_temp);
                              stage10++; }
                          else
                            stage10 = 1;
                            break;
                  default: switch (stage10) {
                             case 4: sprintf(buffer_temp,
"220%3d%3d%3d221%3d%3d%3d222%3d%3d%3d223%3d%3d%3d\n", 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0);
                                     strcat(buffer10, buffer_temp);
                                     break;
                             case 5: sprintf(buffer_temp,
"221%3d%3d%3d222%3d%3d%3d223%3d%3d%3d\n", 0, 0, 0, 0, 0, 0, 0, 0, 0);
                                     strcat(buffer10, buffer_temp);
                                     break;
                             case 6: sprintf(buffer_temp, "222%3d%3d%3d223%3d%3d%3d\n", 0, 0,
0, 0, 0, 0);
                                     strcat(buffer10, buffer_temp);
                                     break;
                             case 7: sprintf(buffer_temp, "223%3d%3d%3d\n", 0, 0, 0);
                                     strcat(buffer10, buffer_temp);
                                     break;
                             default: stage10 = 1;
                                      break;
                             }
                  while ((handle = sopen(append_file_name_to_path("XPLEX10.TXT"),
O_CREAT | O_WRONLY, SH_DENYNO, S_IWRITE)) == FALSE);
                  length = filelength(handle) + (long)XPLEX10_LENGTH + 1;
                  while (locking(handle, LK_NBRLCK, length) != -1);
                  lseek(handle, 0L, SEEK_END);
                  write(handle, buffer10, XPLEX10_LENGTH);
                  lseek(handle, 0L, SEEK_SET);
                  while (locking(handle, LK_UNLCK, length) != -1);
                  close(handle);
                  stage10 = 1;
                  sprintf(buffer10, "%s%3d%d%s%5u%2d%3d%3d%3d", time_stmp, count, $1,
node, $4, $2, $5, $6, $7);
                  stage10++;
                  break;
                  }
```

```
            }

          | device_type CODE ORIGINATING_NODE_NO COMMENT number
LC_COMMENT
                                number COMMENT average_ peak_ last_
          {
          if (strcmp(tme, previous_time) == 0)
            count = count + 1;
          else
            { strcpy(previous_time, tme);
            count = 1; }
          switch(stage10) {
            case 4: if ($5 == 0)
                      stage10++;
                    else
                      { switch($5) {
                          case 1: sprintf(buffer_temp, "223%3d%3d%3d", 0, 0, 0);
                                  strcat(buffer10, buffer_temp);
                                  stage10 = 6;
                                  break;
                          case 2: sprintf(buffer_temp, "222%3d%3d%3d223%3d%3d%3d", 0, 0,
0, 0, 0, 0);
                                  strcat(buffer10, buffer_temp);
                                  stage10 = 7;
                                  break;
                          case 3: sprintf(buffer_temp,
"221%3d%3d%3d222%3d%3d%3d223%3d%3d%3d", 0, 0, 0, 0, 0, 0, 0, 0, 0);
                                  strcat(buffer10, buffer_temp);
                                  sprintf(buffer_temp, "%2d%1d%3d%3d%3d\n", $2, $5, $9,
$10, $11);
                                  strcat(buffer10, buffer_temp);
                                  while ((handle =
sopen(append_file_name_to_path("XPLEX10.TXT"), O_CREAT | O_WRONLY, SH_DENYNO,
S_IWRITE)) == FALSE);
                                  length = filelength(handle) + (long)XPLEX10_LENGTH + 1;
                                  while (locking(handle, LK_NBRLCK, length) != -1);
                                  lseek(handle, 0L, SEEK_END);
                                  write(handle, buffer10, XPLEX10_LENGTH);
                                  lseek(handle, 0L, SEEK_SET);
                                  while (locking(handle, LK_UNLCK, length) != -1);
                                  close(handle);
                                  stage10 = 1;
                                  break;
                          }

                      }
                    if ($5 != 3)
                      { sprintf(buffer_temp, "%2d%1d%3d%3d%3d", $2, $5, $9, $10, $11);
                        strcat(buffer10, buffer_temp); }
                    break;
            case 5: if ($5 == 1)
                      stage10++;
                    else
                      { switch($5) {
                          case 2: sprintf(buffer_temp, "223%3d%3d%3d", 0, 0, 0);
                                  strcat(buffer10, buffer_temp);
                                  stage10 = 7;
                                  break;
                          case 3:
```

```
                                    sprintf(buffer_temp, "222%3d%3d%3d223%3d%3d%3d", 0, 0,
0, 0, 0, 0);
                                    strcat(buffer10, buffer_temp);
                                    sprintf(buffer_temp, "%2d%1d%3d%3d%3d\n", $2, $5, $9,
$10, $11);
                                    strcat(buffer10, buffer_temp);
                                    while ((handle =
sopen(append_file_name_to_path("XPLEX10.TXT"), O_CREAT | O_WRONLY, SH_DENYNO,
S_IWRITE)) == FALSE);
                                    length = filelength(handle) + (long)XPLEX10_LENGTH + 1;
                                    while (locking(handle, LK_NBRLCK, length) != -1);
                                    lseek(handle, 0L, SEEK_END);
                                    write(handle, buffer10, XPLEX10_LENGTH);
                                    lseek(handle, 0L, SEEK_SET);
                                    while (locking(handle, LK_UNLCK, length) != -1);
                                    close(handle);
                                    stage10 = 1;
                                    break;
                                }
                            }
                            if ($5 != 3)
                              { sprintf(buffer_temp, "%2d%1d%3d%3d%3d", $2, $5, $9, $10, $11);
                                strcat(buffer10, buffer_temp); }
                            break;
                    case 6: if ($5 == 2)
                              stage10++;
                            else
                              { switch($5) {
                                  case 3: sprintf(buffer_temp, "223%3d%3d%3d", 0, 0, 0);
                                    strcat(buffer10, buffer_temp);
                                    sprintf(buffer_temp, "%2d%1d%3d%3d%3d\n", $2, $5, $9,
$10, $11);
                                    strcat(buffer10, buffer_temp);
                                    while ((handle =
sopen(append_file_name_to_path("XPLEX10.TXT"), O_CREAT | O_WRONLY, SH_DENYNO,
S_IWRITE)) == FALSE);
                                    length = filelength(handle) + (long)XPLEX10_LENGTH + 1;
                                    while (locking(handle, LK_NBRLCK, length) != -1);
                                    lseek(handle, 0L, SEEK_END);
                                    write(handle, buffer10, XPLEX10_LENGTH);
                                    lseek(handle, 0L, SEEK_SET);
                                    while (locking(handle, LK_UNLCK, length) != -1);
                                    close(handle);
                                    stage10 = 1;
                                    break;
                                }
                              }
                            if ($5 != 3)
                              { sprintf(buffer_temp, "%2d%1d%3d%3d%3d", $2, $5, $9, $10, $11);
                                strcat(buffer10, buffer_temp); }
                            break;
                    case 7: if ($5 == 3)
                              { switch($5) {
                                  case 3: sprintf(buffer_temp, "%2d%1d%3d%3d%3d\n", $2, $5, $9,
$10, $11);
                                    strcat(buffer10, buffer_temp);
                                    while ((handle =
sopen(append_file_name_to_path("XPLEX10.TXT"), O_CREAT | O_WRONLY, SH_DENYNO,
S_IWRITE)) == FALSE);
                                    length = filelength(handle) + (long)XPLEX10_LENGTH + 1;
```

```
                                while (locking(handle, LK_NBRLCK, length) != -1);
                                lseek(handle, 0L, SEEK_END);
                                write(handle, buffer10, XPLEX10_LENGTH);
                                lseek(handle, 0L, SEEK_SET);
                                while (locking(handle, LK_UNLCK, length) != -1);
                                close(handle);
                                stage10 = 1;
                                break;
                            }
                        }
                    break;
                }
            }

            | device_type CODE ORIGINATING_NODE_NO DIAL_STRING NEW_LINE

            {
            if (strcmp(tme, previous_time) == 0)
              count = count + 1;
            else { strcpy(previous_time, tme);
              count = 1; }
            sprintf(buffer, "%s%3d%d%2d%s%s\n", time_stmp, count, $1, $2, node, dl_string);
            while ((handle = sopen(append_file_name_to_path("XPLEX04.TXT"), O_CREAT |
O_WRONLY, SH_DENYNO, S_IWRITE)) == FALSE);
            string_length = strlen(file_buffer);
            length = filelength(handle) + (long)string_length + 1;
            while (locking(handle, LK_NBRLCK, length) != -1);
            lseek(handle, 0L, SEEK_END);
            write(handle, file_buffer, string_length);
            lseek(handle, 0L, SEEK_SET);
            while (locking(handle, LK_UNLCK, length) != -1);
            close(handle);
            }

            | device_type CODE ORIGINATING_NODE_NO COMMENT NEW_LINE

            {
            if (strcmp(tme, previous_time) == 0)
              count = count + 1;
            else { strcpy(previous_time, tme);
              count = 1; }
            sprintf(file_buffer, "%s%3d%d%2d%s\n", time_stmp, count, $1, $2, node);
            while ((handle = sopen(append_file_name_to_path("XPLEX02.TXT"), O_CREAT |
O_WRONLY, SH_DENYNO, S_IWRITE)) == FALSE);
            length = filelength(handle) + (long)XPLEX02_LENGTH + 1;
            while (locking(handle, LK_NBRLCK, length) != -1);
            lseek(handle, 0L, SEEK_END);
            write(handle, file_buffer, XPLEX02_LENGTH);
            lseek(handle, 0L, SEEK_SET);
            while (locking(handle, LK_UNLCK, length) != -1);
            close(handle);
            }

            | device_type CODE LINK_ADDRESS COMMENT NEW_LINE

            {
            if (strcmp(tme, previous_time)==0)
              count = count + 1;
            else { strcpy(previous_time, tme);
              count = 1; }
```

```
            sprintf(file_buffer, "%s%3d%d%2d%s%s\n", time_stmp, count, $1, $2, node, link);
            while ((handle = sopen(append_file_name_to_path("XPLEX03.TXT"), O_CREAT |
O_WRONLY, SH_DENYNO, S_IWRITE)) == FALSE);
            length = filelength(handle) + (long)XPLEX03_LENGTH + 1;
            while (locking(handle, LK_NBRLCK, length) != -1);
            lseek(handle, 0L, SEEK_END);
            write(handle, file_buffer, XPLEX03_LENGTH);
            lseek(handle, 0L, SEEK_SET);
            while (locking(handle, LK_UNLCK, length) != -1);
            close(handle);
            }

            | device_type CODE LINK_ADDRESS minutes_ transmit_
                        receive_ NEW_LINE

            {
            if (strcmp(tme, previous_time)==0)
              count = count + 1;
            else { strcpy(previous_time, tme);
              count = 1; }
            switch (stage11) {
              case 1: if ($2 == 23)
                        { sprintf(buffer11, "%s%3d%d%s%s%5u%2d%5u%5u", time_stmp, count,
$1, node, link, $4, $2, $5, $6);
                          stage11++; }
                      break;

              case 2: if ($2 == 24)
                        { sprintf(buffer_temp, "%2d%5u%5u", $2, $5, $6);
                          strcat(buffer11, buffer_temp);
                          stage11++; }
                      else
                        stage11 = 1;
                      break;
              case 3: if ($2 == 25)
                        { sprintf(buffer_temp, "%2d%5u%5u", $2, $5, $6);
                          strcat(buffer11, buffer_temp);
                          stage11++; }
                      else
                        stage11 = 1;
                      break;
              case 4: if ($2 == 26)
                        { sprintf(buffer_temp, "%2d%5u%5u", $2, $5, $6);
                          strcat(buffer11, buffer_temp);
                          stage11++; }
                        else
                          stage11 = 1;
                      break;
              default: stage11 = 1;
                       break;
              }
            }

            | device_type CODE LINK_ADDRESS minutes_ average_ peak_ last_

            {
            if (strcmp(tme, previous_time)==0)
              count = count + 1;
            else
              { strcpy(previous_time, tme);
```

```
                count = 1; }
              switch (stage11) {
                case 5: if ($2 == 27)
                          { sprintf(buffer_temp, "%2d%3u%3u%3u", $2, $5, $6, $7);
                            strcat(buffer11, buffer_temp);
                            stage11++; }
                        else
                          stage11 = 1;
                        break;
                case 6: if ($2 == 28)
                          { sprintf(buffer_temp, "%2d%3u%3u%3u", $2, $5, $6, $7);
                            strcat(buffer11, buffer_temp);
                            stage11++; }
                          else
                            stage11 = 1;
                        break;
                case 7: if ($2 == 29)
                          { sprintf(buffer_temp, "%2d%3u%3u%3u\n", $2, $5, $6, $7);
                            strcat(buffer11, buffer_temp);
                            while ((handle = sopen(append_file_name_to_path("XPLEX11.TXT"),
O_CREAT | O_WRONLY, SH_DENYNO, S_IWRITE)) == FALSE);
                            length = filelength(handle) + (long)XPLEX11_LENGTH + 1;
                            while (locking(handle, LK_NBRLCK, length) != -1);
                            lseek(handle, 0L, SEEK_END);
                            write(handle, buffer11, XPLEX11_LENGTH);
                            lseek(handle, 0L, SEEK_SET);
                            while (locking(handle, LK_UNLCK, length) != -1);
                            close(handle);
                            stage11 = 1;
                          }
                        else
                          stage11 = 1;
                        break;
                default: stage11 = 1;
                         break;
              }
            }

            | device_type CODE ADDRESS COMMENT NEW_LINE

            {
            if (strcmp(tme, previous_time) == 0)
              count = count + 1;
            else
              { strcpy(previous_time, tme);
                count = 1; }
            sprintf(file_buffer, "%s%3d%d%2d%s%s%s\n", time_stmp, count, $1, $2, node, link,
channel);
            while ((handle = sopen(append_file_name_to_path("XPLEX05.TXT"), O_CREAT |
O_WRONLY, SH_DENYNO, S_IWRITE)) == FALSE);
            length = filelength(handle) + (long)XPLEX05_LENGTH + 1;
            while (locking(handle, LK_NBRLCK, length) != -1);
            lseek(handle, 0L, SEEK_END);
            write(handle, file_buffer, XPLEX05_LENGTH);
            lseek(handle, 0L, SEEK_SET);
            while (locking(handle, LK_UNLCK, length) != -1);
            close(handle);
            }

            | device_type CODE ADDRESS transmit_ receive_ COMMENT
```

```
                        COMMENT NEW_LINE

                {
                if (strcmp(tme, previous_time) == 0)
                  count = count + 1;
                else
                  { strcpy(previous_time, tme);
                    count = 1; }
                sprintf(file_buffer, "%s%3d%d%2d%s%s%s%5u%5u\n", time_stmp, count, $1, $2,
node, link, channel, $4, $5);
                while ((handle = sopen(append_file_name_to_path("XPLEX08.TXT"), O_CREAT |
O_WRONLY, SH_DENYNO, S_IWRITE)) == FALSE);
                length = filelength(handle) + (long)XPLEX08_LENGTH + 1;
                while (locking(handle, LK_NBRLCK, length) != -1);
                lseek(handle, 0L, SEEK_END);
                write(handle, file_buffer, XPLEX08_LENGTH);
                lseek(handle, 0L, SEEK_SET);
                while (locking(handle, LK_UNLCK, length) != -1);
                close(handle);
                }

                | device_type CODE START_UP_MESSAGE transmit_ receive_ COMMENT
                                COMMENT NEW_LINE

                {
                if (strcmp(tme, previous_time) == 0)
                  count = count + 1;
                else
                  { strcpy(previous_time, tme);
                    count = 1; }
                sprintf(file_buffer, "%s%3d%d%2d%s%s%s%5u%5u\n", time_stmp, count, $1, $2,
node, link, channel, $4, $5);
                while ((handle = sopen(append_file_name_to_path("XPLEX08.TXT"), O_CREAT |
O_WRONLY, SH_DENYNO, S_IWRITE)) == FALSE);
                length = filelength(handle) + (long)XPLEX08_LENGTH + 1;
                while (locking(handle, LK_NBRLCK, length) !=-1);
                lseek(handle, 0L, SEEK_END);
                write(handle, file_buffer, XPLEX08_LENGTH);
                lseek(handle, 0L, SEEK_SET);
                while (locking(handle, LK_UNLCK, length) != -1);
                close(handle);
                }

                | device_type CODE ADDRESS number COMMENT NEW_LINE

                { if (strcmp(tme, previous_time) == 0)
                    count = count + 1;
                else
                  { strcpy(previous_time, tme);
                    count = 1; }
                sprintf(file_buffer, "%s%3d%d%2d%s%s%s%2d\n", time_stmp, count, $1, $2, node,
link, channel, $4);
                while ((handle = sopen(append_file_name_to_path("XPLEX07.TXT"), O_CREAT |
O_WRONLY, SH_DENYNO, S_IWRITE)) == FALSE);
                length = filelength(handle) + (long)XPLEX07_LENGTH + 1;
                while (locking(handle, LK_NBRLCK, length) != -1);
                lseek(handle, 0L, SEEK_END);
                write(handle, file_buffer, XPLEX07_LENGTH);
                lseek(handle, 0L, SEEK_SET);
                while (locking(handle, LK_UNLCK, length) != -1);
```

```
                close(handle);
                }

                | device_type CODE addresses COMMENT NEW_LINE

                { if (strcmp(tme, previous_time) == 0)
                    count = count + 1;
                else
                  { strcpy(previous_time, tme);
                    count = 1; }
                sprintf(file_buffer, "%s%3d%d%2d%s%s%s%s%s%s\n", time_stmp, count, $1, $2,
orig_node, orig_link, orig_channel, node, link, channel);
                while ((handle = sopen(append_file_name_to_path("XPLEX12.TXT"), O_CREAT |
O_WRONLY, SH_DENYNO, S_IWRITE)) == FALSE);
                length = filelength(handle) + (long)XPLEX12_LENGTH + 1;
                while (locking(handle, LK_NBRLCK, length) != -1);
                lseek(handle, 0L, SEEK_END);
                write(handle, file_buffer, XPLEX12_LENGTH);
                lseek(handle, 0L, SEEK_SET);
                while (locking(handle, LK_UNLCK, length) != -1);
                close(handle);
                }

                | device_type CODE ADDRESS DIAL_STRING COMMENT NEW_LINE

                {
                if (strcmp(tme, previous_time) == 0)
                  count = count + 1;
                else
                  { strcpy(previous_time, tme);
                    count = 1; }
                sprintf(file_buffer, "%s%3d%d%2d%s%s%s%s\n", time_stmp, count, $1, $2, node,
link, channel, dl_string);
                while ((handle = sopen(append_file_name_to_path("XPLEX06.TXT"), O_CREAT |
O_WRONLY, SH_DENYNO, S_IWRITE)) == FALSE);
                string_length = strlen(file_buffer);
                length = filelength(handle) + (long)string_length + 1;
                while (locking(handle, LK_NBRLCK, length) != -1);
                lseek(handle, 0L, SEEK_END);
                write(handle, file_buffer, string_length);
                lseek(handle, 0L, SEEK_SET);
                while (locking(handle, LK_UNLCK, length) != -1);
                close(handle);
                }

                | device_type CODE ADDRESS DIAL_STRING NEW_LINE

                {
                if (strcmp(tme, previous_time) == 0)
                  count = count + 1;
                else { strcpy(previous_time, tme);
                  count = 1; }
                sprintf(file_buffer, "%s%3d%d%2d%s%s%s%s\n", time_stmp, count, $1, $2, node,
link, channel, dl_string);
                while ((handle = sopen(append_file_name_to_path("XPLEX06.TXT"), O_CREAT |
O_WRONLY, SH_DENYNO, S_IWRITE)) == FALSE);
                string_length = strlen(file_buffer);
                length = filelength(handle) + (long)string_length + 1;
                while (locking(handle, LK_NBRLCK, length) != -1);
                lseek(handle, 0L, SEEK_END);
```

```
                write(handle, file_buffer, string_length);
                lseek(handle, 0L, SEEK_SET);
                while (locking(handle, LK_UNLCK, length) != -1);
                close(handle);
                }
                ;

minutes_  :  COMMENT number COMMENT  { $$ = $2;}
          ;

average_  :  number AVE_COMMENT  { $$ = $1;}
          ;

peak_  :  number COMMENT  { $$ = $1;}
       ;

last_  :  number NEW_LINE     { $$ = $1;}
       ;

transmit_  :  number TRANS_COMMENT { $$ = $1;}
           ;

receive_  :  number  { $$ = $1;}
          ;

number        :  NUMB  { $$ = $1; }
       | DEVICE_TYPE     { $$ = $1; }
       | ORIGINATING_NODE_NO       { $$ = $1; }
       ;

addresses  :  CALL_ADDRESS
           | CONTROL_ADDRESS
           ;

device_type  :DEVICE_TYPE

            {
            switch($1) {
              case 331: $$ = 1;
                        break;
              case 440: $$ = 2;
                        break;
              case 445: $$ = 3;
                        break;
              case 330: $$ = 4;
                        break;
              }
            }
            ;

%%

#include "LEXYY.C"

yyerror(s)
  char *s;
{
  fprintf(stderr, "Error: %s\n", s);
}
```

```
yywrap()
{
  return 1;
}


char *
append_file_name_to_path(file_name)
  char file_name[12];
{
  char new_path[268];
  strcpy(new_path, path_name);
  strcat(new_path, file_name);
  return(new_path);
}


main()
{
  memset(input_stackptr.inputChar, '\0', 256);
  input_stackptr.top = 0;
  while((buffer10 = (char *)malloc(XPLEX10_LENGTH)) == NULL);
  while((buffer11 = (char *)malloc(XPLEX11_LENGTH)) == NULL);
  stage10 = 1;
  stage11 = 1;
  count = 0;
  fp = fopen("EVENTCFG.DAT", "r");
  fscanf(fp, "%d%d%d%s", &speed, &port_number, &word_length, path_name);
  fclose(fp);
  open_com(speed, port_number, word_length);
  strcpy(previous_time, "0\0");
  yyparse();
  close_com();
  free(buffer);
}
```

```
2       1       8
D:\TELLABS\TXTFILES\

/* The EVENTCFG.DAT file contains the configuration of:

    the COM port for the event log

      ARGUMENTS   : speed          - an integer value to indicate baudrate
                                   for the com ports.
                                   0 = 300 baud.       - allowable but not
                                   1 = 600 baud.         recommended.
                                   2 = 1200 baud.      - highly recommended.
                                   3 = 2400 baud.      - the rest are not
                                   4 = 4800 baud.      recommended for
                                   5 = 9600 baud.        during peak loads.
                                   6 = 19200 baud.
                                   7 = 38400 baud.

                    port_number    - 1 = COM1.     - default.
                                     2 = COM2.

                    word_length    - 7 = 7 bits.
                                     8 = 8 bits.   - setup for Tellabs.

    the destination directory for the database text files

      ARGUMENTS   : drive: \ directory path\


EXAMPLE:

2       1       8
D:\TELLABS\TXTFILES\

Note: the fields for the COM port are tab delimited (or space) and the path
      name is on the next line. The path name must also exist before trying to
      write to the directory or else it will fail.
*/
```

```
2       1       8
D:\TELLABS\TXTFILES\

/* The EVENTCFG.DAT file contains the configuration of:

    the COM port for the event log

      ARGUMENTS   : speed          - an integer value to indicate baudrate
                                   for the com ports.
                                   0 = 300 baud.       - allowable but not
                                   1 = 600 baud.         recommended.
                                   2 = 1200 baud.      - highly recommended.
                                   3 = 2400 baud.      - the rest are not
                                   4 = 4800 baud.      recommended for
                                   5 = 9600 baud.        during peak loads.
                                   6 = 19200 baud.
                                   7 = 38400 baud.

                  port_number      - 1 = COM1.       - default.
                                   2 = COM2.

                  word_length      - 7 = 7 bits.
                                   8 = 8 bits.  - setup for Tellabs.

    the destination directory for the database text files

      ARGUMENTS   :  drive: \ directory path\


EXAMPLE:

2       1       8
D:\TELLABS\TXTFILES\

Note: the fields for the COM port are tab delimited (or space) and the path
      name is on the next line. The path name must also exist before trying to
      write to the directory or else it will fail.
*/
```

```
# include "h:\users\lbw\c600\include\stdio.h"
# define U(x) x
# define NLSTATE yyprevious=YYNEWLINE
# define BEGIN yybgin = yysvec + 1 +
# define INITIAL 0
# define YYLERR yysvec
# define YYSTATE (yyestate-yysvec-1)
# define YYOPTIM 1
# define YYLMAX BUFSIZ
# define output(c) putc(c,yyout)
# define input() (((yytchar=yysptr>yysbuf?U(*--
yysptr):getc(yyin))==10?(yylineno++,yytchar):yytchar)==EOF?0:yytchar)
# define unput(c) {yytchar= (c);if(yytchar=='\n')yylineno--;*yysptr++=yytchar;}
# define yymore() (yymorfg=1)
# define ECHO fprintf(yyout, "%s",yytext)
# define REJECT { nstr = yyreject(); goto yyfussy;}
int yyleng; extern char yytext[];
int yymorfg;
extern char *yysptr, yysbuf[];
int yytchar;
FILE *yyin = {stdin}, *yyout = {stdout};
extern int yylineno;
struct yysvf {
        struct yywork *yystoff;
        struct yysvf *yyother;
        int *yystops;};
struct yysvf *yyestate;
extern struct yysvf yysvec[], *yybgin;



char pop()

{

        if (input_stackptr.top == 0)

          printf("Trying to pop a character off the stack which doesn't exist\n");

        else

          return(input_stackptr.inputChar[-- input_stackptr.top]);

}



push(c)

        char c;

{

        if (input_stackptr.top == 256)

          printf("Stack overflow in unput()\n");

        else

          input_stackptr.inputChar[input_stackptr.top ++] = c;
```

```
}

int empty()
{
        if (input_stackptr.top == 0)
          return(TRUE);
        else
          return(FALSE);
}

# undef input
input()
{
char return_value;
        if (empty())
          {
            return_value = read_com();
            return(return_value);
          }
        else
        {
          return_value = pop();
          return(return_value);
        }
}

# undef unput
unput(c)
char c;
{
        push(c);
```

```
}


# define YYNEWLINE 10
yylex(){
int nstr; extern int yyprevious;
while((nstr = yylook()) >= 0)
yyfussy: switch(nstr){
case 0:
if(yywrap()) return(0); break;
case 1:
                    ;
break;
case 2:
                    { printf("%s", yytext); return(NEW_LINE); }
break;
case 3:
                    { printf("%s", yytext); }
break;
case 4:
                    { printf("%s", yytext); }
break;
case 5:
          { printf("%s", yytext); return(LC_COMMENT); }
break;
case 6:
                    { printf("%s", yytext); return(TRANS_COMMENT); }
break;
case 7:
                    { printf("%s", yytext); return(COMMENT); }
break;
case 8:
              { printf("%s", yytext); return(COMMENT); }
break;
case 9:
                    { printf("%s", yytext); return(COMMENT); }
break;
case 10:
                    { printf("%s", yytext); return(COMMENT); }
break;
case 11:
              { printf("%s", yytext); return(COMMENT); }
break;
case 12:
              { printf("%s", yytext); return(COMMENT); }
break;
case 13:
                    { printf("%s", yytext); return(COMMENT); }
break;
case 14:
              { printf("%s", yytext); return(COMMENT); }
break;
case 15:
              { printf("%s", yytext); return(COMMENT); }
break;
case 16:
                    { printf("%s", yytext); return(COMMENT); }
break;
```

```
case 17:
            { printf("%s", yytext); return(COMMENT); }
break;
case 18:
                    { printf("%s", yytext); return(COMMENT); }
break;
case 19:
                    { printf("%s", yytext); return(COMMENT); }
break;
case 20:
                    { printf("%s", yytext); return(AVE_COMMENT); }
break;
case 21:
                    { printf("%s", yytext); return(COMMENT); }
break;
case 22:
                    { printf("%s", yytext); return(TRANS_COMMENT); }
break;
case 23:
       { printf("%s", yytext); }
break;
case 24:
            { printf("%s", yytext); }
break;
case 25:
       { printf("%s", yytext); }
break;
case 26:
            { printf("%s", yytext); }
break;
case 27:
       { printf("%s", yytext); }
break;
case 28:
       { printf("%s", yytext); }
break;
case 29:
                    { printf("%s", yytext); return(COMMENT); }
break;
case 30:
            { printf("%s", yytext); return(COMMENT); }
break;
case 31:
            { printf("%s", yytext); return(COMMENT); }
break;
case 32:
            { printf("%s", yytext); return(COMMENT); }
break;
case 33:
            { printf("%s", yytext); return(COMMENT); }
break;
case 34:
            { printf("%s", yytext); return(COMMENT); }
break;
case 35:
       { printf("%s", yytext); return(COMMENT); }
break;
case 36:
            { printf("%s", yytext); return(COMMENT); }
break;
```

```
case 37:
                        { printf("%s", yytext); return(COMMENT); }
break;
case 38:
                { printf("%s", yytext); return(COMMENT); }
break;
case 39:
                { printf("%s", yytext); return(COMMENT); }
break;
case 40:
                        { printf("%s", yytext); return(COMMENT); }
break;
case 41:
        { printf("%s", yytext); return(COMMENT); }
break;
case 42:
                { printf("%s", yytext); return(COMMENT); }
break;
case 43:
                { printf("%s", yytext); return(COMMENT); }
break;
case 44:
        { printf("%s", yytext); return(COMMENT); }
break;
case 45:
{ printf("%s", yytext); return(COMMENT); }
break;
case 46:
        { printf("%s", yytext); return(BLANK_COMMENT); }
break;
case 47:
        { printf("%s", yytext); return(BLANK_COMMENT); }
break;
case 48:
        { printf("%s", yytext); return(BLANK_COMMENT); }
break;
case 49:
   { printf("%s", yytext); return(BLANK_COMMENT); }
break;
case 50:
                { printf("%s", yytext); return(BLANK_COMMENT); }
break;
case 51:
   { printf("%s", yytext); return(BLANK_COMMENT); }
break;
case 52:
        { printf("%s", yytext); }
break;
case 53:
{ printf("%s", yytext);

                                *(yytext + 2) = '\0';

                                *(yytext + 5) = '\0';

                                if (atoi(yytext + 6) >= 80)

                                        strcpy(dte, "19");

                                else
```

```
                                strcpy(dte, "20");
                        strcat(dte, (yytext + 6));
                        strncat(dte, (yytext), 2);
                        strncat(dte, (yytext + 3), 2);
                        return(DATE); }
break;
case 54:
        { printf("%s", yytext);
                        *(yytext + 2) = '\0';
                        strcpy(tme, yytext);
                        strncat(tme, (yytext + 3), 2);
                        return(TIME); }
break;
case 55:
{ printf("%s", yytext);
                yylval = atoi(yytext);
                  return(DEVICE_TYPE); }
break;
case 56:
{ printf("%s", yytext);
                        *(yytext + 3) = '\0';
                        strcpy(code_event, (yytext + 1));
                        strncat(code_event, (yytext + 4), 2);
                        switch (atoi(code_event)) {
                                case 100:       yylval = 1;
                                                break;
                                case 200:       yylval = 2;
                                                break;
                                case 300:       yylval = 3;
                                                break;
                                case 402:       yylval = 4;
                                                break;
                                case 403:       yylval = 5;
                                                break;
```

```
case 404:       yylval = 6;
                break;
case 405:       yylval = 7;
                break;
case 406:       yylval = 8;
                break;
case 407:       yylval = 9;
                break;
case 500:       yylval = 10;
                break;
case 600:       yylval = 11;
                break;
case 700:       yylval = 12;
                break;
case 800:       yylval = 13;
                break;
case 900:       yylval = 14;
                break;
case 1000:      yylval = 15;
                break;
case 1100:      yylval = 16;
                break;
case 1200:      yylval = 17;
                break;
case 1300:      yylval = 18;
                break;
case 1401:      yylval = 19;
                break;
case 1402:      yylval = 20;
                break;
```

```
case 1403:      yylval = 21;
                break;
case 1404:      yylval = 22;
                break;
case 1405:      yylval = 23;
                break;
case 1406:      yylval = 24;
                break;
case 1407:      yylval = 25;
                break;
case 1408:      yylval = 26;
                break;
case 1409:      yylval = 27;
                break;
case 1410:      yylval = 28;
                break;
case 1411:      yylval = 29;
                break;
case 1500:      yylval = 30;
                break;
case 1501:      yylval = 31;
                break;
case 1510:      yylval = 32;
                break;
case 1511:      yylval = 33;
                break;
case 1600:      yylval = 34;
                break;
case 1700:      yylval = 35;
                break;
```

```
case 1800:      yylval = 36;
                break;
case 1801:      yylval = 37;
                break;
case 1812:      yylval = 38;
                break;
case 1814:      yylval = 39;
                break;
case 1817:      yylval = 40;
                break;
case 1819:      yylval = 41;
                break;
case 1901:      yylval = 42;
                break;
case 1902:      yylval = 43;
                break;
case 1903:      yylval = 44;
                break;
case 1904:      yylval = 45;
                break;
case 1905:      yylval = 46;
                break;
case 1906:      yylval = 47;
                break;
case 1907:      yylval = 48;
                break;
case 1908:      yylval = 49;
                break;
case 1909:      yylval = 50;
                break;
```

```
                    case 1910:      yylval = 51;
                                    break;
                    case 1911:      yylval = 52;
                                    break;
                    case 1912:      yylval = 53;
                                    break;
                    case 1913:      yylval = 54;
                                    break;
                    case 1914:      yylval = 55;
                                    break;
                    case 1915:      yylval = 56;
                                    break;
                    case 1916:      yylval = 57;
                                    break;
                    case 1917:      yylval = 58;
                                    break;
                    case 1918:      yylval = 59;
                                    break;
                    case 1919:      yylval = 60;
                                    break;
                    case 2000:      yylval = 61;
                                    break;
                    case 2100:      yylval = 62;
                                    break;
                    case 2200:      yylval = 63;
                                    break;
                }
                return(CODE); }
break;
case 57:
{ printf("%s", yytext);
        yylval = atoi(yytext);
```

```
                return(ORIGINATING_NODE_NO); }
break;
case 58:
    {
                                        printf("%s", yytext);
                                        *(yytext + 3) = '\0';
                                        *(yytext + 7) = '\0';
                                        *(yytext + 11) = '\0';
                                        *(yytext + 15) = '\0';
                                        strcpy(orig_node, yytext);
                                    strcpy(orig_link, (yytext + 4));
                                strcpy(orig_channel, (yytext + 8));
                                        strcpy(node, yytext + 12);
                                        strcpy(link, "CON");   /* CONTROL */
                                        strcpy(channel, "---");/* blank */
                                    return(CONTROL_ADDRESS); }
break;
case 59:
      { printf("%s", yytext);
                        *(yytext + 3) = '\0';
                        strcpy(node, yytext);
                    strcpy(link, "---");        /* blank */
                        strcpy(channel, (yytext + 5)); /* X.25 */
                        return(START_UP_MESSAGE); }
break;
case 60:
  { printf("%s", yytext);
                        *(yytext + 3) = '\0';
                        strcpy(node, yytext);
                    strcpy(link, (yytext + 4));
                    return(LINK_ADDRESS); }
break;
case 61:
     { printf("%s", yytext);
                                    *(yytext + 3) = '\0';
                                *(yytext + 7) = '\0';
```

```
                                    strcpy(node, yytext);
                                    strcpy(link, (yytext + 4));
                                    strcpy(channel, (yytext + 8));
                                        return(ADDRESS); }
break;
case 62:
     {
                                        printf("%s", yytext);
                                        *(yytext + 3) = '\0';
                                    *(yytext + 7) = '\0';
                                        *(yytext + 11) = '\0';
                                        *(yytext + 15) = '\0';
                                        *(yytext + 19) = '\0';
                                    strcpy(orig_node, yytext);
                                    strcpy(orig_link, (yytext + 4));
                                    strcpy(orig_channel, (yytext + 8));
                                        strcpy(node, (yytext + 12));
                                    strcpy(link, (yytext + 16));
                                    strcpy(channel, (yytext + 20));
                                        return(CALL_ADDRESS); }
break;
case 63:
     { printf("%s", yytext);
               *(yytext + (yyleng - 1)) = '\0';
               strcpy(dl_string, (yytext + 1));
               return(DIAL_STRING); }
break;
case 64:
       { printf("%s", yytext);
                     numb = atoi(yytext);
                   yylval = atoi(yytext);
                     return(NUMB); }
break;
case 65:
            { printf("%s", yytext); numb = 65536; return(NUMB); }
break;
case -1:
```

```
break;
default:
fprintf(yyout,"bad switch yylook %d",nstr);
} return(0); }
/* end of yylex */



int yyvstop[] = {
0,

3,
0,

2,
0,

1,
0,

3,
0,

64,
0,

64,
0,

64,
0,

64,
0,

63,
0,

64,
0,

64,
0,

64,
0,

7,
0,

6,
0,

20,
0,

21,
0,

22,
```

0,

19,
0,

57,
0,

57,
64,
0,

55,
57,
64,
0,

40,
0,

13,
0,

37,
0,

64,
0,

10,
0,

29,
0,

16,
0,

18,
0,

65,
0,

54,
0,

4,
0,

9,
0,

56,
0,

34,
0,

14,

0,

60,
0,

8,
0,

33,
0,

38,
0,

32,
39,
0,

52,
0,

43,
0,

12,
0,

53,
0,

59,
0,

15,
0,

26,
0,

42,
0,

17,
0,

30,
0,

31,
0,

61,
0,

36,
0,

11,
0,

```
41,
0,

24,
0,

35,
0,

25,
0,

5,
0,

50,
0,

47,
0,

28,
0,

27,
0,

48,
0,

23,
0,

46,
0,

44,
0,

62,
0,

58,
0,

45,
0,

49,
0,

51,
0,
0};
# define YYTYPE int
struct yywork { YYTYPE verify, advance; } yycrank[] = {
0,0,    0,0,    0,0,    0,0,
0,0,    0,0,    0,0,    0,0,
0,0,    0,0,    1,3,    1,4,
0,0,    3,3,    1,5,    7,7,
```

6,3, 0,0, 0,0, 0,0,
0,0, 0,0, 0,0, 7,7,
0,0, 0,0, 0,0, 0,0,
0,0, 0,0, 0,0, 0,0,
0,0, 1,6, 0,0, 1,7,
3,3, 0,0, 0,0, 6,3,
0,0, 1,8, 0,0, 1,9,
9,40, 0,0, 0,0, 0,0,
7,34, 1,10, 1,11, 1,11,
1,12, 1,13, 1,11, 1,11,
1,11, 1,11, 1,11, 1,14,
19,53, 12,43, 7,7, 7,7,
13,44, 0,0, 1,15, 1,16,
1,17, 1,18, 1,19, 0,0,
6,30, 1,20, 1,21, 30,74,
31,75, 1,22, 32,76, 1,23,
1,24, 21,55, 33,77, 1,25,
1,26, 1,27, 1,28, 6,31,
40,84, 1,29, 8,35, 6,32,
8,36, 8,37, 0,0, 0,0,
0,0, 0,0, 0,0, 0,0,
0,0, 0,0, 0,0, 8,38,
0,0, 0,0, 0,0, 0,0,
0,0, 26,67, 8,39, 0,0,
0,0, 0,0, 22,59, 0,0,
6,33, 10,41, 10,41, 10,41,
10,41, 10,41, 10,41, 10,41,
10,41, 10,41, 10,41, 11,42,
11,42, 11,42, 11,42, 11,42,
11,42, 11,42, 11,42, 11,42,
11,42, 14,45, 14,45, 14,45,
14,45, 14,45, 14,45, 14,45,
14,45, 14,45, 14,45, 15,46,
16,47, 17,48, 18,51, 20,54,
22,60, 24,65, 18,52, 25,66,
17,49, 21,56, 21,57, 21,58,
23,61, 26,68, 27,70, 17,50,
23,62, 28,72, 29,73, 35,78,
36,79, 37,80, 39,83, 38,81,
46,91, 47,92, 23,63, 48,93,
49,94, 27,71, 50,95, 51,96,
23,64, 38,82, 41,85, 41,86,
41,86, 41,86, 41,86, 41,86,
41,86, 41,86, 41,86, 41,86,
41,86, 41,87, 52,97, 53,98,
54,99, 55,100, 26,69, 42,88,
42,88, 42,88, 42,88, 42,88,
42,88, 42,88, 42,88, 42,88,
42,88, 43,88, 43,89, 43,88,
43,88, 43,88, 43,88, 43,88,
43,88, 43,88, 43,88, 44,89,
44,88, 44,88, 44,88, 44,88,
44,89, 44,88, 44,88, 44,88,
44,88, 45,90, 45,90, 45,90,
45,90, 45,90, 45,90, 45,90,
45,90, 45,90, 45,90, 56,101,
57,102, 58,103, 59,105, 60,106,
61,107, 62,108, 63,110, 64,111,
62,109, 65,112, 66,113, 67,114,
68,115, 69,116, 70,117, 71,118,

72,119. 73,120, 78,121, 79,122,
58,104. 80,123, 81,124, 82,125,
83,126. 84,127, 85,128, 85,128,
85,128. 85,128, 85,128, 85,128,
85,128. 85,128, 85,128, 85,128,
86,129. 87,130, 87,130, 87,130,
87,130. 87,130, 87,130, 87,130,
87,130. 87,130, 87,130, 88,129,
88,131, 88,131, 88,131, 88,131,
88,131. 88,131, 88,131, 88,131,
88,131, 88,131, 90,132, 91,133,
92,134. 93,135, 94,136, 95,138,
96,139, 97,140, 94,137, 98,141,

100,142.	101,143,	102,144,	103,145,
104,146.	105,147,	106,148,	107,149,
108,150.	111,151,	112,152,	113,153,
114,154.	115,155,	116,156,	117,157,
118,158.	119,159,	120,160,	121,161,
122,162.	123,163,	124,164,	125,165,
126,166.	127,167,	128,168,	128,168,
128,168.	128,168,	128,168,	128,168,
128,168.	128,168,	128,168,	128,168,
129,169.	133,173,	129,170,	129,170,
129,170.	129,170,	129,170,	129,170,
129,170.	129,170,	129,170,	129,170,
130,171.	130,171,	130,171,	130,171,
130,171.	130,171,	130,171,	130,171,
130,171.	130,171,	131,131,	131,131,
131,131,	131,131,	131,131,	131,131,
131,131,	131,131,	131,131,	131,131,
132,172.	132,172,	132,172,	132,172,
132,172.	132,172,	132,172,	132,172,
132,172.	132,172,	134,174,	135,175,
136,177.	137,178,	138,179,	139,180,
140,183.	141,184,	142,185,	143,186,
144,187.	145,188,	146,189,	147,190,
149,191.	150,192,	151,193,	152,194,
153,195.	154,196,	155,197,	157,198,
158,199,	159,200,	160,201,	161,202,
162,203.	163,204,	164,205,	165,206,
166,207.	168,208,	169,209,	170,210,
170,210.	170,210,	170,210,	170,210,
170,210.	170,210,	170,210,	170,210,
170,210.	172,211,	172,211,	172,211,
172,211.	172,211,	172,211,	172,211,
172,211.	172,211,	172,211,	173,212,
174,213.	175,214,	177,215,	178,216.
180,217.	181,218,	182,219,	183,220,
184,221.	185,222,	186,223,	187,224,
188,225.	189,226,	190,227,	191,228,
192,229.	139,181,	193,230.	194,231.
195,232.	196,233,	139,182.	197,234.
198,235.	199,236,	200,237.	201,238.
202,239.	203,240,	135,176,	204,241.
205,242.	206,243,	207,244.	208,245.
208,245.	208,245,	208,245.	208,245.
208,245.	208,245,	208,245.	208,245.
208,245.	209,246,	210,247.	210,247.
210,247.	210,247,	210,247.	210,247.
210,247.	210,247,	210,247.	210,247.

212,248, 213,249, 214,250, 215,251,
216,252, 217,253, 218,254, 219,255,
220,256, 221,257, 222,258, 223,259,
224,260, 225,261, 226,262, 227,263,
228,264, 229,265, 231,266, 232,267,
233,268, 234,269, 236,270, 237,271,
238,272, 239,273, 240,274, 241,275,
242,276, 243,277, 244,278, 245,279,
245,279, 245,279, 245,279, 245,279,
245,279, 245,279, 245,279, 245,279,
245,279, 246,280, 246,280, 246,280,
246,280, 246,280, 246,280, 246,280,
246,280, 246,280, 246,280, 247,281,
248,282, 249,283, 250,284, 252,285,
253,286, 254,287, 255,288, 256,289,
257,290, 258,291, 260,292, 261,293,
263,294, 264,295, 266,296, 267,297,
268,298, 270,299, 273,300, 274,301,
275,302, 276,303, 277,304, 278,305,
281,306, 281,306, 281,306, 281,306,
281,306, 281,306, 281,306, 281,306,
281,306, 281,306, 282,307, 283,308,
284,309, 285,310, 286,311, 288,312,
289,313, 291,314, 292,315, 293,316,
294,317, 295,318, 297,319, 298,320,
299,321, 300,322, 301,323, 302,324,
303,325, 304,326, 305,327, 306,328,
306,328, 306,328, 306,328, 306,328,
306,328, 306,328, 306,328, 306,328,
306,328, 307,329, 309,330, 310,331,
312,332, 313,333, 314,334, 315,335,
316,336, 317,337, 318,338, 319,339,
320,340, 321,341, 322,342, 323,343,
324,344, 325,345, 326,346, 327,347,
328,348, 328,348, 328,348, 328,348,
328,348, 328,348, 328,348, 328,348,
328,348, 328,348, 329,349, 330,350,
331,351, 332,352, 333,353, 334,354,
336,355, 337,356, 338,357, 339,358,
340,359, 341,360, 342,361, 343,362,
344,363, 345,364, 346,365, 347,366,
348,367, 349,368, 350,369, 351,370,
352,371, 354,372, 355,373, 356,374,
357,375, 358,376, 359,377, 360,378,
361,379, 362,380, 363,381, 364,382,
365,383, 366,384, 367,367, 368,386,
370,387, 371,388, 372,389, 348,367,
373,390, 374,391, 375,392, 376,393,
377,394, 378,395, 379,396, 380,397,
381,398, 382,399, 383,400, 384,401,
386,403, 387,404, 388,405, 389,406,
390,407, 367,367, 385,402, 385,402,
385,402, 385,402, 385,402, 385,402,
385,402, 385,402, 385,402, 385,402,
391,408, 392,409, 393,410, 394,411,
395,412, 367,385, 367,385, 367,385,
367,385, 367,385, 367,385, 367,385,
367,385, 367,385, 367,385, 396,413,
397,414, 398,415, 399,416, 400,417,
401,418, 402,419, 402,419, 402,419,

```
402,419,	402,419,	402,419,	402,419,
402,419,	402,419,	402,419,	404,420,
405,421,	407,422,	408,423,	410,424,
412,425,	413,426,	414,427,	415,428,
416,429,	417,430,	418,431,	419,432,
420,433,	421,434,	422,435,	424,436,
425,437,	426,438,	429,439,	430,440,
431,441,	432,442,	432,442,	432,442,
432,442,	432,442,	432,442,	432,442,
432,442,	432,442,	432,442,	433,444,
434,445,	435,446,	438,447,	440,448,
441,449,	443,451,	445,452,	446,453,
432,443,	442,450,	442,450,	442,450,
442,450,	442,450,	442,450,	442,450,
442,450,	442,450,	442,450,	447,454,
448,455,	449,456,	450,457,	450,457,
450,457,	450,457,	450,457,	450,457,
450,457,	450,457,	450,457,	450,457,
451,458,	452,459,	453,460,	454,461,
456,462,	457,463,	458,464,	459,465,
461,466,	462,467,	463,468,	463,468,
463,468,	463,468,	463,468,	463,468,
463,468,	463,468,	463,468,	463,468,
464,469,	465,470,	466,471,	467,472,
468,473,	468,473,	468,473,	468,473,
468,473,	468,473,	468,473,	468,473,
468,473,	468,473,	469,474,	470,475,
471,476,	472,477,	473,478,	473,478,
473,478,	473,478,	473,478,	473,478,
473,478,	473,478,	473,478,	473,478,
474,479,	475,480,	476,481,	477,482,
481,483,	482,484,	0,0,	0,0,
0,0};
struct yysvf yysvec[] = {
0,	0,	0,
yycrank+1,	0,	0,
yycrank+0,	yysvec+1,	0,
yycrank+4,	0,	yyvstop+1,
yycrank+0,	0,	yyvstop+3,
yycrank+0,	0,	yyvstop+5,
yycrank+7,	0,	yyvstop+7,
yycrank+-14,	0,	0,
yycrank+25,	0,	0,
yycrank+2,	0,	0,
yycrank+69,	0,	yyvstop+9,
yycrank+79,	0,	yyvstop+11,
yycrank+10,	yysvec+11,	yyvstop+13,
yycrank+12,	yysvec+11,	yyvstop+15,
yycrank+89,	0,	0,
yycrank+47,	0,	0,
yycrank+31,	0,	0,
yycrank+52,	0,	0,
yycrank+53,	0,	0,
yycrank+28,	0,	0,
yycrank+40,	0,	0,
yycrank+49,	0,	0,
yycrank+47,	0,	0,
yycrank+63,	0,	0,
yycrank+41,	0,	0,
yycrank+35,	0,	0,
```

```
yycrank+77,   0,          0,
yycrank+57,   0,          0,
yycrank+55,   0,          0,
yycrank+54,   0,          0,
yycrank+43,   0,          0,
yycrank+44,   0,          0,
yycrank+46,   0,          0,
yycrank+50,   0,          0,
yycrank+0,    0,          yyvstop+17,
yycrank+67,   0,          0,
yycrank+71,   0,          0,
yycrank+72,   0,          0,
yycrank+70,   0,          0,
yycrank+60,   0,          0,
yycrank+46,   0,          0,
yycrank+135,  0,          0,
yycrank+151,  yysvec+41,  yyvstop+19,
yycrank+161,  yysvec+41,  yyvstop+21,
yycrank+171,  yysvec+41,  yyvstop+23,
yycrank+181,  0,          0,
yycrank+63,   0,          0,
yycrank+71,   0,          0,
yycrank+67,   0,          0,
yycrank+79,   0,          0,
yycrank+61,   0,          0,
yycrank+63,   0,          0,
yycrank+79,   0,          0,
yycrank+125,  0,          0,
yycrank+84,   0,          0,
yycrank+127,  0,          0,
yycrank+131,  0,          0,
yycrank+128,  0,          0,
yycrank+142,  0,          0,
yycrank+210,  0,          0,
yycrank+133,  0,          0,
yycrank+135,  0,          0,
yycrank+129,  0,          0,
yycrank+130,  0,          0,
yycrank+138,  0,          0,
yycrank+148,  0,          0,
yycrank+218,  0,          yyvstop+25,
yycrank+181,  0,          0,
yycrank+187,  0,          0,
yycrank+138,  0,          0,
yycrank+145,  0,          0,
yycrank+223,  0,          yyvstop+27,
yycrank+149,  0,          0,
yycrank+149,  0,          0,
yycrank+0,    0,          yyvstop+29,
yycrank+0,    0,          yyvstop+31,
yycrank+0,    0,          yyvstop+33,
yycrank+0,    0,          yyvstop+35,
yycrank+149,  0,          0,
yycrank+151,  0,          0,
yycrank+145,  0,          0,
yycrank+146,  0,          0,
yycrank+149,  0,          0,
yycrank+157,  0,          0,
yycrank+223,  0,          0,
yycrank+218,  0,          0,
```

```
yycrank+229,  0,            yyvstop+37,
yycrank+229,  0,            0,
yycrank+240,  0,            yyvstop+39,
yycrank+0,    yysvec+88,    yyvstop+42,
yycrank+252,  0,            0,
yycrank+194,  0,            0,
yycrank+198,  0,            0,
yycrank+193,  0,            0,
yycrank+192,  0,            0,
yycrank+193,  0,            0,
yycrank+207,  0,            0,
yycrank+189,  0,            0,
yycrank+193,  0,            0,
yycrank+0,    0,            yyvstop+46,
yycrank+194,  0,            0,
yycrank+208,  0,            0,
yycrank+199,  0,            0,
yycrank+200,  0,            0,
yycrank+215,  0,            0,
yycrank+247,  0,            0,
yycrank+207,  0,            0,
yycrank+214,  0,            0,
yycrank+197,  0,            0,
yycrank+0,    0,            yyvstop+48,
yycrank+0,    0,            yyvstop+50,
yycrank+219,  0,            0,
yycrank+204,  0,            0,
yycrank+234,  0,            0,
yycrank+206,  0,            0,
yycrank+243,  0,            0,
yycrank+276,  0,            0,
yycrank+222,  0,            0,
yycrank+239,  0,            0,
yycrank+215,  0,            0,
yycrank+225,  0,            0,
yycrank+222,  0,            0,
yycrank+220,  0,            0,
yycrank+232,  0,            0,
yycrank+211,  0,            0,
yycrank+222,  0,            0,
yycrank+222,  0,            0,
yycrank+291,  0,            0,
yycrank+286,  0,            0,
yycrank+298,  0,            0,
yycrank+308,  0,            0,
yycrank+318,  0,            yyvstop+52,
yycrank+328,  0,            0,
yycrank+235,  0,            0,
yycrank+285,  0,            0,
yycrank+355,  0,            yyvstop+54,
yycrank+278,  0,            0,
yycrank+292,  0,            0,
yycrank+274,  0,            0,
yycrank+359,  0,            yyvstop+56,
yycrank+287,  0,            0,
yycrank+296,  0,            0,
yycrank+297,  0,            0,
yycrank+292,  0,            0,
yycrank+281,  0,            0,
yycrank+288,  0,            0,
```

```
yycrank+290,  0,          0,
yycrank+282,  0,          0,
yycrank+0,    0,          yyvstop+58,
yycrank+368,  0,          0,
yycrank+290,  0,          0,
yycrank+301,  0,          0,
yycrank+306,  0,          0,
yycrank+288,  0,          0,
yycrank+308,  0,          0,
yycrank+338,  0,          0,
yycrank+0,    0,          yyvstop+60,
yycrank+375,  0,          0,
yycrank+292,  0,          0,
yycrank+298,  0,          0,
yycrank+290,  0,          0,
yycrank+301,  0,          0,
yycrank+380,  0,          0,
yycrank+381,  0,          0,
yycrank+303,  0,          0,
yycrank+318,  0,          0,
yycrank+305,  0,          0,
yycrank+0,    0,          yyvstop+62,
yycrank+370,  0,          0,
yycrank+368,  0,          0,
yycrank+371,  0,          0,
yycrank+0,    0,          yyvstop+64,
yycrank+381,  0,          0,
yycrank+334,  0,          0,
yycrank+326,  0,          0,
yycrank+374,  0,          0,
yycrank+0,    0,          yyvstop+66,
yycrank+341,  0,          0,
yycrank+344,  0,          0,
yycrank+0,    0,          yyvstop+68,
yycrank+368,  0,          0,
yycrank+348,  0,          0,
yycrank+332,  0,          0,
yycrank+337,  0,          0,
yycrank+339,  0,          0,
yycrank+340,  0,          0,
yycrank+353,  0,          0,
yycrank+336,  0,          0,
yycrank+340,  0,          0,
yycrank+348,  0,          0,
yycrank+352,  0,          0,
yycrank+377,  0,          0,
yycrank+342,  0,          0,
yycrank+344,  0,          0,
yycrank+343,  0,          0,
yycrank+355,  0,          0,
yycrank+352,  0,          0,
yycrank+397,  0,          0,
yycrank+379,  0,          0,
yycrank+360,  0,          0,
yycrank+347,  0,          0,
yycrank+366,  0,          0,
yycrank+363,  0,          0,
yycrank+385,  0,          0,
yycrank+395,  0,          0,
yycrank+358,  0,          0,
```

```
yycrank+365,  0,        0,
yycrank+355,  0,        0,
yycrank+427,  0,        0,
yycrank+432,  0,        0,
yycrank+438,  0,        0,
yycrank+0,    0,        yyvstop+70,
yycrank+381,  0,        0,
yycrank+457,  0,        0,
yycrank+387,  0,        0,
yycrank+391,  0,        0,
yycrank+384,  0,        0,
yycrank+396,  0,        0,
yycrank+387,  0,        0,
yycrank+406,  0,        0,
yycrank+407,  0,        0,
yycrank+404,  0,        0,
yycrank+405,  0,        0,
yycrank+399,  0,        0,
yycrank+403,  0,        0,
yycrank+401,  0,        0,
yycrank+410,  0,        0,
yycrank+409,  0,        0,
yycrank+401,  0,        0,
yycrank+406,  0,        0,
yycrank+0,    0,        yyvstop+72,
yycrank+403,  0,        0,
yycrank+407,  0,        0,
yycrank+415,  0,        0,
yycrank+428,  0,        0,
yycrank+0,    0,        yyvstop+74,
yycrank+410,  0,        0,
yycrank+409,  0,        0,
yycrank+406,  0,        0,
yycrank+406,  0,        0,
yycrank+417,  0,        0,
yycrank+418,  0,        0,
yycrank+417,  0,        0,
yycrank+493,  0,        0,
yycrank+416,  0,        0,
yycrank+479,  0,        0,
yycrank+489,  0,        0,
yycrank+501,  0,        yyvstop+76,
yycrank+432,  0,        0,
yycrank+434,  0,        0,
yycrank+440,  0,        0,
yycrank+0,    0,        yyvstop+78,
yycrank+450,  0,        0,
yycrank+442,  0,        0,
yycrank+452,  0,        0,
yycrank+445,  0,        0,
yycrank+439,  0,        0,
yycrank+441,  0,        0,
yycrank+442,  0,        0,
yycrank+0,    0,        yyvstop+80,
yycrank+460,  0,        0,
yycrank+458,  0,        0,
yycrank+0,    0,        yyvstop+82,
yycrank+459,  0,        0,
yycrank+445,  0,        0,
yycrank+0,    0,        yyvstop+84,
```

```
yycrank+448,  0,          0,
yycrank+458,  0,          0,
yycrank+449,  0,          0,
yycrank+0,    0,          yyvstop+87,
yycrank+460,  0,          0,
yycrank+0,    0,          yyvstop+89,
yycrank+0,    0,          yyvstop+91,
yycrank+450,  0,          0,
yycrank+458,  0,          0,
yycrank+458,  0,          0,
yycrank+537,  0,          0,
yycrank+502,  0,          0,
yycrank+539,  0,          0,
yycrank+0,    0,          yyvstop+93,
yycrank+0,    0,          yyvstop+95,
yycrank+524,  0,          0,
yycrank+468,  0,          0,
yycrank+542,  0,          0,
yycrank+474,  0,          0,
yycrank+471,  0,          0,
yycrank+479,  0,          0,
yycrank+0,    0,          yyvstop+97,
yycrank+555,  0,          0,
yycrank+483,  0,          0,
yycrank+0,    0,          yyvstop+99,
yycrank+549,  0,          0,
yycrank+482,  0,          0,
yycrank+475,  0,          0,
yycrank+478,  0,          0,
yycrank+561,  0,          0,
yycrank+0,    0,          yyvstop+101,
yycrank+472,  0,          0,
yycrank+555,  0,          0,
yycrank+474,  0,          0,
yycrank+483,  0,          0,
yycrank+497,  0,          0,
yycrank+492,  0,          0,
yycrank+530,  0,          0,
yycrank+496,  0,          0,
yycrank+534,  0,          0,
yycrank+555,  0,          0,
yycrank+516,  0,          0,
yycrank+0,    0,          yyvstop+103,
yycrank+513,  0,          0,
yycrank+500,  0,          0,
yycrank+0,    0,          yyvstop+105,
yycrank+549,  0,          0,
yycrank+506,  0,          0,
yycrank+498,  0,          0,
yycrank+518,  0,          0,
yycrank+519,  0,          0,
yycrank+589,  0,          0,
yycrank+552,  0,          0,
yycrank+526,  0,          0,
yycrank+504,  0,          0,
yycrank+528,  0,          0,
yycrank+529,  0,          0,
yycrank+527,  0,          0,
yycrank+596,  0,          0,
yycrank+532,  0,          0,
```

```
yycrank+515,  0,          0,
yycrank+526,  0,          0,
yycrank+584,  0,          0,
yycrank+526,  0,          0,
yycrank+544,  0,          0,
yycrank+604,  0,          0,
yycrank+541,  0,          0,
yycrank+536,  0,          0,
yycrank+598,  0,          0,
yycrank+0,    0,          yyvstop+107,
yycrank+616,  0,          0,
yycrank+564,  0,          0,
yycrank+539,  0,          0,
yycrank+535,  0,          0,
yycrank+603,  0,          0,
yycrank+537,  0,          0,
yycrank+538,  0,          0,
yycrank+623,  0,          0,
yycrank+588,  0,          0,
yycrank+552,  0,          0,
yycrank+559,  0,          0,
yycrank+544,  0,          0,
yycrank+651,  0,          yyvstop+109,
yycrank+556,  0,          0,
yycrank+546,  0,          0,
yycrank+543,  0,          0,
yycrank+563,  0,          0,
yycrank+0,    0,          yyvstop+111,
yycrank+617,  0,          0,
yycrank+598,  0,          0,
yycrank+551,  0,          0,
yycrank+551,  0,          0,
yycrank+564,  0,          0,
yycrank+622,  0,          0,
yycrank+566,  0,          0,
yycrank+567,  0,          0,
yycrank+594,  0,          0,
yycrank+563,  0,          0,
yycrank+567,  0,          0,
yycrank+565,  0,          0,
yycrank+578,  0,          0,
yycrank+669,  0,          0,
yycrank+568,  0,          0,
yycrank+0,    0,          yyvstop+113,
yycrank+631,  0,          0,
yycrank+582,  0,          0,
yycrank+634,  0,          0,
yycrank+587,  0,          0,
yycrank+580,  0,          0,
yycrank+576,  0,          0,
yycrank+576,  0,          0,
yycrank+640,  0,          0,
yycrank+578,  0,          0,
yycrank+579,  0,          0,
yycrank+574,  0,          0,
yycrank+573,  0,          0,
yycrank+576,  0,          0,
yycrank+584,  0,          0,
yycrank+584,  0,          0,
yycrank+654,  0,          0,
```

```
yycrank+586,  0,        0,
yycrank+649,  0,        0,
yycrank+591,  0,        0,
yycrank+658,  0,        0,
yycrank+584,  0,        0,
yycrank+604,  0,        0,
yycrank+613,  0,        0,
yycrank+604,  0,        0,
yycrank+674,  0,        0,
yycrank+606,  0,        0,
yycrank+617,  0,        0,
yycrank+612,  0,        0,
yycrank+619,  0,        0,
yycrank+616,  0,        0,
yycrank+621,  0,        0,
yycrank+622,  0,        0,
yycrank+685,  0,        0,
yycrank+0,    0,        yyvstop+115,
yycrank+695,  0,        0,
yycrank+629,  0,        0,
yycrank+0,    0,        yyvstop+117,
yycrank+648,  0,        0,
yycrank+700,  0,        0,
yycrank+0,    0,        yyvstop+119,
yycrank+715,  0,        0,
yycrank+0,    0,        yyvstop+121,
yycrank+716,  0,        0,
yycrank+717,  0,        0,
yycrank+709,  0,        0,
yycrank+710,  0,        0,
yycrank+651,  0,        0,
yycrank+652,  0,        0,
yycrank+644,  0,        0,
yycrank+708,  0,        0,
yycrank+708,  0,        0,
yycrank+640,  0,        0,
yycrank+655,  0,        0,
yycrank+0,    0,        yyvstop+123,
yycrank+722,  0,        0,
yycrank+723,  0,        0,
yycrank+682,  0,        0,
yycrank+0,    0,        yyvstop+125,
yycrank+0,    0,        yyvstop+127,
yycrank+721,  0,        0,
yycrank+664,  0,        0,
yycrank+663,  0,        0,
yycrank+717,  0,        0,
yycrank+734,  0,        0,
yycrank+667,  0,        0,
yycrank+663,  0,        0,
yycrank+0,    0,        yyvstop+129,
yycrank+0,    0,        yyvstop+131,
yycrank+660,  0,        0,
yycrank+0,    0,        yyvstop+133,
yycrank+663,  0,        0,
yycrank+681,  0,        0,
yycrank+737,  0,        0,
yycrank+702,  0,        0,
yycrank+0,    0,        yyvstop+135,
yycrank+750,  0,        0,
```

```
yycrank+686,  0,          0,
yycrank+694,  0,          0,
yycrank+755,  0,          0,
yycrank+681,  0,          0,
yycrank+750,  0,          0,
yycrank+730,  0,          0,
yycrank+740,  0,          0,
yycrank+701,  0,          0,
yycrank+697,  0,          0,
yycrank+0,    0,          yyvstop+137,
yycrank+780,  0,          0,
yycrank+767,  0,          0,
yycrank+730,  0,          0,
yycrank+701,  0,          0,
yycrank+0,    0,          yyvstop+139,
yycrank+702,  0,          0,
yycrank+750,  0,          0,
yycrank+770,  0,          0,
yycrank+746,  0,          0,
yycrank+715,  0,          0,
yycrank+725,  0,          0,
yycrank+720,  0,          0,
yycrank+784,  0,          0,
yycrank+763,  0,          0,
yycrank+732,  0,          0,
yycrank+744,  0,          0,
yycrank+745,  0,          0,
yycrank+798,  0,          0,
yycrank+780,  0,          0,
yycrank+743,  0,          0,
yycrank+757,  0,          0,
yycrank+758,  0,          0,
yycrank+0,    0,          yyvstop+141,
yycrank+0,    0,          yyvstop+143,
yycrank+0,    0,          yyvstop+145,
yycrank+819,  0,          0,
yycrank+820,  0,          0,
yycrank+0,    0,          yyvstop+147,
yycrank+0,    0,          yyvstop+149,
0,      0,      0};
struct yywork *yytop = yycrank+861;
struct yysvf *yybgin = yysvec+1;
char yymatch[] = {
00  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
01  ,011 ,01  ,01  ,01  ,01  ,01  ,01  ,
01  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
01  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
011 ,01  ,'"' ,01  ,01  ,01  ,01  ,01  ,
01  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
'0' ,'1' ,'1' ,'1' ,'1' ,'1' ,'1' ,'1' ,
'1' ,'1' ,01  ,01  ,01  ,01  ,01  ,01  ,
01  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
01  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
01  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
01  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
01  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
01  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
0};
```

```
char yyextra[] = {
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0};
#ifndef lint
static  char ncform_sccsid[] = "@(#)ncform 1.6 88/02/08 SMI"; /* from S5R2 1.2 */
#endif

int yylineno =1;
# define YYU(x) x
# define NLSTATE yyprevious=YYNEWLINE
char yytext[YYLMAX];
struct yysvf *yylstate [YYLMAX], **yylsp, **yyolsp;
char yysbuf[YYLMAX];
char *yysptr = yysbuf;
int *yyfnd;
extern struct yysvf *yyestate;
int yyprevious = YYNEWLINE;
yylook(){
        register struct yysvf *yystate, **lsp;
        register struct yywork *yyt;
        struct yysvf *yyz;
        int yych, yyfirst;
        struct yywork *yyr;
# ifdef LEXDEBUG
        int debug;
# endif
        char *yylastch;
        /* start off machines */
# ifdef LEXDEBUG
        debug = 0;
# endif
        yyfirst=1;
        if (!yymorfg)
                yylastch = yytext;
        else {
                yymorfg=0;
                yylastch = yytext+yyleng;
                }
        for(;;){
                lsp = yylstate;
                yyestate = yystate = yybgin;
                if (yyprevious==YYNEWLINE) yystate++;
                for (;;){
# ifdef LEXDEBUG
                        if(debug)fprintf(yyout,"state %d\n",yystate-yysvec-1);
# endif
                        yyt = yystate->yystoff;
                        if(yyt == yycrank && !yyfirst){  /* may not be any transitions */
                                yyz = yystate->yyother;
                                if(yyz == 0)break;
                                if(yyz->yystoff == yycrank)break;
                                }
```

```
                *yylastch++ = yych = input();
                yyfirst=0;
        tryagain:
# ifdef LEXDEBUG
                if(debug){
                        fprintf(yyout,"char ");
                        allprint(yych);
                        putchar('\n');
                        }
# endif
                yyr = yyt;
                if ( (int)yyt > (int)yycrank){
                        yyt = yyr + yych;
                        if (yyt <= yytop && yyt->verify+yysvec == yystate){
                                if(yyt->advance+yysvec == YYLERR)       /* error
transitions */
                                        {unput(*--yylastch);break;}
                                *lsp++ = yystate = yyt->advance+yysvec;
                                goto contin;
                                }
                        }
# ifdef YYOPTIM
                else if((int)yyt < (int)yycrank) {              /* r < yycrank */
                        yyt = yyr = yycrank+(yycrank-yyt);
# ifdef LEXDEBUG
                        if(debug)fprintf(yyout,"compressed state\n");
# endif
                        yyt = yyt + yych;
                        if(yyt <= yytop && yyt->verify+yysvec == yystate){
                                if(yyt->advance+yysvec == YYLERR)       /* error
transitions */
                                        {unput(*--yylastch);break;}
                                *lsp++ = yystate = yyt->advance+yysvec;
                                goto contin;
                                }
                        yyt = yyr + YYU(yymatch[ych]);
# ifdef LEXDEBUG
                        if(debug){
                                fprintf(yyout,"try fall back character ");
                                allprint(YYU(yymatch[ych]));
                                putchar('\n');
                                }
# endif
                        if(yyt <= yytop && yyt->verify+yysvec == yystate){
                                if(yyt->advance+yysvec == YYLERR)       /* error
transition */
                                        {unput(*--yylastch);break;}
                                *lsp++ = yystate = yyt->advance+yysvec;
                                goto contin;
                                }
                        }
                if ((yystate = yystate->yyother) && (yyt= yystate->yystoff) != yycrank){
# ifdef LEXDEBUG
                        if(debug)fprintf(yyout,"fall back to state %d\n",yystate-yysvec-1);
# endif
                        goto tryagain;
                        }
# endif
                else
                        {unput(*--yylastch);break;}
```

```
                contin:
# ifdef LEXDEBUG
                        if(debug){
                                fprintf(yyout,"state %d char ",yystate-yysvec-1);
                                allprint(yych);
                                putchar('\n');
                                }
# endif
                        ;
                        }
# ifdef LEXDEBUG
                if(debug){
                        fprintf(yyout,"stopped at %d with ",*(lsp-1)-yysvec-1);
                        allprint(yych);
                        putchar('\n');
                        }
# endif
                while (lsp-- > yylstate){
                        *yylastch-- = 0;
                        if (*lsp != 0 && (yyfnd= (*lsp)->yystops) && *yyfnd > 0){
                                yyolsp = lsp;
                                if(yyextra[*yyfnd]){            /* must backup */
                                        while(yyback((*lsp)->yystops,-*yyfnd) != 1 && lsp >
yylstate){
                                                lsp--;
                                                unput(*yylastch--);
                                                }
                                        }
                                yyprevious = YYU(*yylastch);
                                yylsp = lsp;
                                yyleng = yylastch-yytext+1;
                                yytext[yyleng] = 0;
# ifdef LEXDEBUG
                                if(debug){
                                        fprintf(yyout,"\nmatch ");
                                        sprint(yytext);
                                        fprintf(yyout," action %d\n",*yyfnd);
                                        }
# endif
                                return(*yyfnd++);
                                }
                        unput(*yylastch);
                        }
                if (yytext[0] == 0  /* && feof(yyin) */)
                        {
                        yysptr=yysbuf;
                        return(0);
                        }
                yyprevious = yytext[0] = input();
                if (yyprevious>0)
                        output(yyprevious);
                yylastch=yytext;
# ifdef LEXDEBUG
                if(debug)putchar('\n');
# endif
                }
        }
yyback(p, m)
        int *p;
{
```

```
if (p==0) return(0);
while (*p)
        {
        if (*p++ == m)
                return(1);
        }
return(0);
}
        /* the following are only used in the lex library */
yyinput(){
        return(input());
        }
yyoutput(c)
  int c; {
        output(c);
        }
yyunput(c)
   int c; {
        unput(c);
        }
```

```
#ifndef lint
static char yysccsid[] = "@(#)yaccpar     1.7 (Berkeley) 09/09/90";
#endif
#define YYBYACC 1
#line 2 "testya.y"
# include <h:\users\lbw\c600\include\stdio.h>
# include <h:\users\lbw\c600\include\string.h>
# include <h:\users\lbw\c600\include\io.h>
# include <h:\users\lbw\c600\include\sys\locking.h>
# include <h:\users\lbw\c600\include\fcntl.h>
# include <h:\users\lbw\c600\include\process.h>
# include <h:\users\lbw\c600\include\share.h>
# include <h:\users\lbw\c600\include\graph.h>
# include <h:\users\lbw\c600\include\malloc.h>
# include <h:\users\lbw\c600\include\dos.h>
# include <h:\users\lbw\c600\include\errno.h>
# include "event.h"

FILE *fp;
#line 21 "y_tab.c"
#define DEVICE_TYPE 257
#define MAJR_CODE 258
#define MINOR_CODE 259
#define ORIGINATING_NODE_NO 260
#define ADDRESS 261
#define COMMENT 262
#define OPTIONAL_EXTENSION 263
#define DESCRIPTOR 264
#define LINK_ADDRESS 265
#define CODE 266
#define DATE 267
#define TIME 268
#define NUMB 269
#define NEW_LINE 270
#define DIAL_STRING 271
#define CONTROL_ADDRESS 272
#define START_UP_MESSAGE 273
#define AVE_COMMENT 274
#define TRANS_COMMENT 275
#define CALL_ADDRESS 276
#define BLANK_COMMENT 277
#define LC_COMMENT 278
#define YYERRCODE 256
short yylhs[] = {                                        -1,
    0,    0,    2,    0,    3,    1,    4,    4,    5,    5,
    5,    5,    5,    5,    5,    5,    5,    5,    5,    5,
    5,    5,    5,    7,    8,    9,   10,   12,   13,   11,
   11,   11,   14,   14,    6,
};
short yylen[] = {                                         2,
    1,    2,    0,    4,    3,    2,    1,    2,    2,    7,
   11,    5,    5,    5,    7,    7,    5,    8,    8,    6,
    5,    6,    5,    3,    2,    2,    2,    2,    1,    1,
    1,    1,    1,    1,    1,
};
short yydefred[] = {                                      0,
    0,    0,    0,    1,    0,    3,    0,    2,   35,    0,
    0,    7,    0,    0,    5,    9,    8,    0,    0,    0,
    0,    0,   34,    0,   33,    0,    0,    0,    0,   31,
   32,    0,   30,    0,    0,    0,    0,    0,    0,    0,
```

```
   0,  13,   0,  12,   0,   0,  17,   0,  23,   0,
  28,  29,   0,  14,   0,   0,   0,   0,   0,  21,
  24,   0,   0,   0,  25,  22,  20,   0,   0,   0,
   0,   0,  10,   0,  26,   0,  16,  15,   0,   0,
  27,  18,  19,   0,   0,  11,
};
short yydgoto[] = {                                      3,
   4,  14,   5,  11,  12,  13,  29,  45,  63,  73,
  52,  36,  53,  26,
};
short yysindex[] = {                                  -210,
-231, -217, -215,   0, -252,   0, -209,   0,   0, -201,
-252,   0, -206, -210,   0,   0,   0, -250, -215, -224,
-226, -194,   0, -202,   0, -192, -220, -199, -202,   0,
   0, -198,   0, -208, -242, -202, -204, -202, -200, -202,
-197,   0, -249,   0, -202, -196,   0, -193,   0, -191,
   0,   0, -188,   0, -186, -202, -256, -202, -182,   0,
   0, -202, -202, -181,   0,   0,   0, -180, -202, -187,
-178, -177,   0, -184,   0, -183,   0,   0, -179, -202,
   0,   0,   0, -202, -202,   0,
};
short yyrindex[] = {                                     0,
   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   1,   0,   0,   0,   0,   0,   0,   0,  88,   0,
   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0,   0,
};
short yygindex[] = {                                    75,
   9,   0,   0,   0,  79,   0,  70, -36, -52, -55,
 -21, -17, -31,   0,
};
#define YYTABLESIZE 268
short yytable[] = {                                    35,
   6,  56,  39,  69,   9,  43,  40,  46,  59,  20,
  21,   8,  61,  77,  22,  55,  57,  65,  51,  50,
  58,  23,  24,  64,  10,  25,  70,   8,  62,  86,
  30,  85,  51,  31,  64,  32,  30,  27,   6,  31,
  72,  74,  33,  84,  34,   1,  28,  74,  33,  42,
   7,   2,  30,  48,  30,  31,   2,  31,  46,  18,
  15,  49,  64,  74,  33,  54,  33,  37,  16,  41,
  44,  47,  60,  68,  51,  61,  66,  65,  67,  71,
  75,  76,  78,  79,  80,  81,  82,   4,  19,  17,
  83,  38,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
```

```
    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
    0,    0,    0,    0,    0,    0,    0,    6,
};
short yycheck[] = {                                     21,
    0,   38,   24,   56,  257,   27,   24,   29,   40,  260,
  261,    3,  262,   69,  265,   37,   38,  274,  275,  262,
   38,  272,  273,   45,  277,  276,   58,   19,  278,   85,
  257,   84,  275,  260,   56,  262,  257,  262,  270,  260,
   62,   63,  269,   80,  271,  256,  271,   69,  269,  270,
  268,  267,  257,  262,  257,  260,  267,  260,   80,  266,
  270,  270,   84,   85,  269,  270,  269,  262,  270,  262,
  270,  270,  270,  262,  275,  262,  270,  274,  270,  262,
  262,  262,  270,  262,  262,  270,  270,    0,   14,   11,
  270,   22,   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,
   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,
   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,
   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,
   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,
   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,
   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,
   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,
   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,
   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,
   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,
   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,
   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,
   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,
   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,
   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,
   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,   -1,
   -1,   -1,   -1,   -1,   -1,   -1,   -1,  267,
};
#define YYFINAL 3
#ifndef YYDEBUG
#define YYDEBUG 0
#endif
#define YYMAXTOKEN 278
#if YYDEBUG
char *yyname[] = {
"end-of-file",0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,"DEVICE_TYPE","MAJR_CODE",
"MINOR_CODE","ORIGINATING_NODE_NO","ADDRESS","COMMENT","OPTIONAL_EXTEN
SION",
"DESCRIPTOR","LINK_ADDRESS","CODE","DATE","TIME","NUMB","NEW_LINE",
"DIAL_STRING","CONTROL_ADDRESS","START_UP_MESSAGE","AVE_COMMENT",
"TRANS_COMMENT","CALL_ADDRESS","BLANK_COMMENT","LC_COMMENT",
};
char *yyrule[] = {
"$accept : event_log",
"event_log : time_based_datagram",
"event_log : event_log time_based_datagram",
"$$1 :",
```

```
"event_log : error NEW_LINE $$1 event_log",
"time_stamp : DATE TIME NEW_LINE",
"time_based_datagram : time_stamp datagram_list",
"datagram_list : xplexer_datagram",
"datagram_list : datagram_list xplexer_datagram",
"xplexer_datagram : BLANK_COMMENT NEW_LINE",
"xplexer_datagram : device_type CODE ORIGINATING_NODE_NO minutes_ average_ peak_
last_",
"xplexer_datagram : device_type CODE ORIGINATING_NODE_NO COMMENT number
LC_COMMENT number COMMENT average_ peak_ last_",
"xplexer_datagram : device_type CODE ORIGINATING_NODE_NO DIAL_STRING
NEW_LINE",
"xplexer_datagram : device_type CODE ORIGINATING_NODE_NO COMMENT NEW_LINE",
"xplexer_datagram : device_type CODE LINK_ADDRESS COMMENT NEW_LINE",
"xplexer_datagram : device_type CODE LINK_ADDRESS minutes_ transmit_ receive_
NEW_LINE",
"xplexer_datagram : device_type CODE LINK_ADDRESS minutes_ average_ peak_ last_",
"xplexer_datagram : device_type CODE ADDRESS COMMENT NEW_LINE",
"xplexer_datagram : device_type CODE ADDRESS transmit_ receive_ COMMENT COMMENT
NEW_LINE",
"xplexer_datagram : device_type CODE START_UP_MESSAGE transmit_ receive_ COMMENT
COMMENT NEW_LINE",
"xplexer_datagram : device_type CODE ADDRESS number COMMENT NEW_LINE",
"xplexer_datagram : device_type CODE addresses COMMENT NEW_LINE",
"xplexer_datagram : device_type CODE ADDRESS DIAL_STRING COMMENT NEW_LINE",
"xplexer_datagram : device_type CODE ADDRESS DIAL_STRING NEW_LINE",
"minutes_ : COMMENT number COMMENT",
"average_ : number AVE_COMMENT",
"peak_ : number COMMENT",
"last_ : number NEW_LINE",
"transmit_ : number TRANS_COMMENT",
"receive_ : number",
"number : NUMB",
"number : DEVICE_TYPE",
"number : ORIGINATING_NODE_NO",
"addresses : CALL_ADDRESS",
"addresses : CONTROL_ADDRESS",
"device_type : DEVICE_TYPE",
};
#endif
#ifndef YYSTYPE
typedef int YYSTYPE;
#endif
#define yyclearin (yychar=(-1))
#define yyerrok (yyerrflag=0)
#ifdef YYSTACKSIZE
#ifndef YYMAXDEPTH
#define YYMAXDEPTH YYSTACKSIZE
#endif
#else
#ifdef YYMAXDEPTH
#define YYSTACKSIZE YYMAXDEPTH
#else
#define YYSTACKSIZE 600
#define YYMAXDEPTH 600
#endif
#endif
int yydebug;
int yynerrs;
int yyerrflag;
```

```
int yychar;
short *yyssp;
YYSTYPE *yyvsp;
YYSTYPE yyval;
YYSTYPE yylval;
short yyss[YYSTACKSIZE];
YYSTYPE yyvs[YYSTACKSIZE];
#define yystacksize YYSTACKSIZE
#line 544 "testya.y"

#include "LEXYY.C"

yyerror(s)
  char *s;
{
  fprintf(stderr, "Error: %s\n", s);
}

yywrap()
{
  return 1;
}


char *
append_file_name_to_path(file_name)
  char file_name[12];
{
  char new_path[268];
  strcpy(new_path, path_name);
  strcat(new_path, file_name);
  return(new_path);
}


main()
{
  memset(input_stackptr.inputChar, '\0', 256);
  input_stackptr.top = 0;
  while((buffer10 = (char *)malloc(XPLEX10_LENGTH)) == NULL);
  while((buffer11 = (char *)malloc(XPLEX11_LENGTH)) == NULL);
  stage10 = 1;
  stage11 = 1;
  count = 0;
  fp = fopen("EVENTCFG.DAT", "r");
  fscanf(fp, "%d%d%d%s", &speed, &port_number, &word_length, path_name);
  fclose(fp);
  open_com(speed, port_number, word_length);
  strcpy(previous_time, "0\0");
  yyparse();
  close_com();
  free(buffer);
}
#line 288 "y_tab.c"
#define YYABORT goto yyabort
#define YYACCEPT goto yyaccept
#define YYERROR goto yyerrlab
int
yyparse()
{
```

```
    register int yym, yyn, yystate;
#if YYDEBUG
    register char *yys;
    extern char *getenv();

    if (yys = getenv("YYDEBUG"))
    {
        yyn = *yys;
        if (yyn >= '0' && yyn <= '9')
            yydebug = yyn - '0';
    }
#endif

    yynerrs = 0;
    yyerrflag = 0;
    yychar = (-1);

    yyssp = yyss;
    yyvsp = yyvs;
    *yyssp = yystate = 0;

yyloop:
    if (yyn = yydefred[yystate]) goto yyreduce;
    if (yychar < 0)
    {
        if ((yychar = yylex()) < 0) yychar = 0;
#if YYDEBUG
        if (yydebug)
        {
            yys = 0;
            if (yychar <= YYMAXTOKEN) yys = yyname[yychar];
            if (!yys) yys = "illegal-symbol";
            printf("yydebug: state %d, reading %d (%s)\n", yystate,
                    yychar, yys);
        }
#endif
    }
    if ((yyn = yysindex[yystate]) && (yyn += yychar) >= 0 &&
            yyn <= YYTABLESIZE && yycheck[yyn] == yychar)
    {
#if YYDEBUG
        if (yydebug)
            printf("yydebug: state %d, shifting to state %d\n",
                    yystate, yytable[yyn]);
#endif
        if (yyssp >= yyss + yystacksize - 1)
        {
            goto yyoverflow;
        }
        *++yyssp = yystate = yytable[yyn];
        *++yyvsp = yylval;
        yychar = (-1);
        if (yyerrflag > 0)  --yyerrflag;
        goto yyloop;
    }
    if ((yyn = yyrindex[yystate]) && (yyn += yychar) >= 0 &&
            yyn <= YYTABLESIZE && yycheck[yyn] == yychar)
    {
        yyn = yytable[yyn];
        goto yyreduce;
```

```
    }
    if (yyerrflag) goto yyinrecovery;
#ifdef lint
    goto yynewerror;
#endif
yynewerror:
    yyerror("syntax error");
#ifdef lint
    goto yyerrlab;
#endif
yyerrlab:
    ++yynerrs;
yyinrecovery:
    if (yyerrflag < 3)
    {
        yyerrflag = 3;
        for (;;)
        {
            if ((yyn = yysindex[*yyssp]) && (yyn += YYERRCODE) >= 0 &&
                    yyn <= YYTABLESIZE && yycheck[yyn] == YYERRCODE)
            {
#if YYDEBUG
                if (yydebug)
                    printf("yydebug: state %d, error recovery shifting\
 to state %d\n", *yyssp, yytable[yyn]);
#endif
                if (yyssp >= yyss + yystacksize - 1)
                {
                    goto yyoverflow;
                }
                *++yyssp = yystate = yytable[yyn];
                *++yyvsp = yylval;
                goto yyloop;
            }
            else
            {
#if YYDEBUG
                if (yydebug)
                    printf("yydebug: error recovery discarding state %d\n",
                            *yyssp);
#endif
                if (yyssp <= yyss) goto yyabort;
                --yyssp;
                --yyvsp;
            }
        }
    }
    else
    {
        if (yychar == 0) goto yyabort;
#if YYDEBUG
        if (yydebug)
        {
            yys = 0;
            if (yychar <= YYMAXTOKEN) yys = yyname[yychar];
            if (!yys) yys = "illegal-symbol";
            printf("yydebug: state %d, error recovery discards token %d (%s)\n",
                    yystate, yychar, yys);
        }
#endif
```

```
        yychar = (-1);
        goto yyloop;
    }
yyreduce:
#if YYDEBUG
    if (yydebug)
        printf("yydebug: state %d, reducing by rule %d (%s)\n",
                yystate, yyn, yyrule[yyn]);
#endif
    yym = yylen[yyn];
    yyval = yyvsp[1-yym];
    switch (yyn)
    {
case 3:
#line 29 "testya.y"
{ yyerrok; }
break;
case 5:
#line 34 "testya.y"
{
            strcpy(time_stmp, dte);
            strcat(time_stmp, tme);
            while ((handle = sopen("DATETIME.TXT", O_CREAT | O_WRONLY, SH_DENYNO,
S_IWRITE)) == FALSE);
            while (locking(handle, LK_NBRLCK, (long) DATE_TIME_LENGTH) != -1);
            lseek(handle, 0L, SEEK_SET);
            write(handle, time_stmp, DATE_TIME_LENGTH);
            lseek(handle, 0L, SEEK_SET);
            while (locking(handle, LK_UNLCK, (long) DATE_TIME_LENGTH) != -1);
            close(handle);
            }
break;
case 10:
#line 56 "testya.y"
{
                if (strcmp(tme, previous_time) == 0)
                  count = count + 1;
                else { strcpy(previous_time, tme);
                  count = 1; }
                switch (stage10) {
                  case 1: if (yyvsp[-5] == 19)
                            { sprintf(buffer10, "%s%3d%d%s%5u%2d%3d%3d%3d", time_stmp,
count, yyvsp[-6], node, yyvsp[-3], yyvsp[-5], yyvsp[-2], yyvsp[-1], yyvsp[0]);
                              stage10++; }
                              break;
                  case 2: if (yyvsp[-5] == 20)
                            { sprintf(buffer_temp, "%2d%3d%3d%3d", yyvsp[-5], yyvsp[-2], yyvsp[-
1], yyvsp[0]);
                              strcat(buffer10, buffer_temp);
                              stage10++; }
                            else
                              stage10 = 1;
                              break;
                  case 3: if (yyvsp[-5] == 21)
                            { sprintf(buffer_temp, "%2d%3d%3d%3d", yyvsp[-5], yyvsp[-2], yyvsp[-
1], yyvsp[0]);
                              strcat(buffer10, buffer_temp);
                              stage10++; }
                            else
                              stage10 = 1;
```

```
                        break;
                default: switch (stage10) {
                        case 4: sprintf(buffer_temp,
"220%3d%3d%3d221%3d%3d%3d222%3d%3d%3d223%3d%3d%3d\n", 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0);
                                strcat(buffer10, buffer_temp);
                                break;
                        case 5: sprintf(buffer_temp,
"221%3d%3d%3d222%3d%3d%3d223%3d%3d%3d\n", 0, 0, 0, 0, 0, 0, 0, 0, 0);
                                strcat(buffer10, buffer_temp);
                                break;
                        case 6: sprintf(buffer_temp, "222%3d%3d%3d223%3d%3d%3d\n", 0, 0,
0, 0, 0, 0);
                                strcat(buffer10, buffer_temp);
                                break;
                        case 7: sprintf(buffer_temp, "223%3d%3d%3d\n", 0, 0, 0);
                                strcat(buffer10, buffer_temp);
                                break;
                        default: stage10 = 1;
                                 break;
                        }
                while ((handle = sopen(append_file_name_to_path("XPLEX10.TXT"),
O_CREAT | O_WRONLY, SH_DENYNO, S_IWRITE)) == FALSE);
                length = filelength(handle) + (long)XPLEX10_LENGTH + 1;
                while (locking(handle, LK_NBRLCK, length) != -1);
                lseek(handle, 0L, SEEK_END);
                write(handle, buffer10, XPLEX10_LENGTH);
                lseek(handle, 0L, SEEK_SET);
                while (locking(handle, LK_UNLCK, length) != -1);
                close(handle);
                stage10 = 1;
                sprintf(buffer10, "%s%3d%d%s%5u%2d%3d%3d%3d", time_stmp, count, yyvsp[-
6], node, yyvsp[-3], yyvsp[-5], yyvsp[-2], yyvsp[-1], yyvsp[0]);
                stage10++;
                break;
                }
              }
break;
case 11:
#line 113 "testya.y"
{
              if (strcmp(tme, previous_time) == 0)
                count = count + 1;
              else
                { strcpy(previous_time, tme);
                count = 1; }
              switch(stage10) {
                case 4: if (yyvsp[-6] == 0)
                          stage10++;
                        else
                          { switch(yyvsp[-6]) {
                              case 1: sprintf(buffer_temp, "223%3d%3d%3d", 0, 0, 0);
                                      strcat(buffer10, buffer_temp);
                                      stage10 = 6;
                                      break;
                              case 2: sprintf(buffer_temp, "222%3d%3d%3d223%3d%3d%3d", 0, 0,
0, 0, 0, 0);
                                      strcat(buffer10, buffer_temp);
                                      stage10 = 7;
                                      break;
```

```
                                case 3: sprintf(buffer_temp,
"221%3d%3d%3d222%3d%3d%3d223%3d%3d%3d", 0, 0, 0, 0, 0, 0, 0, 0, 0);
                                        strcat(buffer10, buffer_temp);
                                        sprintf(buffer_temp, "%2d%1d%3d%3d%3d\n", yyvsp[-9],
yyvsp[-6], yyvsp[-2], yyvsp[-1], yyvsp[0]);
                                        strcat(buffer10, buffer_temp);
                                        while ((handle =
sopen(append_file_name_to_path("XPLEX10.TXT"), O_CREAT | O_WRONLY, SH_DENYNO,
S_IWRITE)) == FALSE);
                                        length = filelength(handle) + (long)XPLEX10_LENGTH + 1;
                                        while (locking(handle, LK_NBRLCK, length) != -1);
                                        lseek(handle, 0L, SEEK_END);
                                        write(handle, buffer10, XPLEX10_LENGTH);
                                        lseek(handle, 0L, SEEK_SET);
                                        while (locking(handle, LK_UNLCK, length) != -1);
                                        close(handle);
                                        stage10 = 1;
                                        break;
                                }

                          }
                         if (yyvsp[-6] != 3)
                          { sprintf(buffer_temp, "%2d%1d%3d%3d%3d", yyvsp[-9], yyvsp[-6],
yyvsp[-2], yyvsp[-1], yyvsp[0]);
                            strcat(buffer10, buffer_temp); }
                         break;
                case 5: if (yyvsp[-6] == 1)
                          stage10++;
                         else
                          { switch(yyvsp[-6]) {
                              case 2: sprintf(buffer_temp, "223%3d%3d%3d", 0, 0, 0);
                                        strcat(buffer10, buffer_temp);
                                        stage10 = 7;
                                        break;
                              case 3:
                                        sprintf(buffer_temp, "222%3d%3d%3d223%3d%3d%3d", 0, 0,
0, 0, 0, 0);
                                        strcat(buffer10, buffer_temp);
                                        sprintf(buffer_temp, "%2d%1d%3d%3d%3d\n", yyvsp[-9],
yyvsp[-6], yyvsp[-2], yyvsp[-1], yyvsp[0]);
                                        strcat(buffer10, buffer_temp);
                                        while ((handle =
sopen(append_file_name_to_path("XPLEX10.TXT"), O_CREAT | O_WRONLY, SH_DENYNO,
S_IWRITE)) == FALSE);
                                        length = filelength(handle) + (long)XPLEX10_LENGTH + 1;
                                        while (locking(handle, LK_NBRLCK, length) != -1);
                                        lseek(handle, 0L, SEEK_END);
                                        write(handle, buffer10, XPLEX10_LENGTH);
                                        lseek(handle, 0L, SEEK_SET);
                                        while (locking(handle, LK_UNLCK, length) != -1);
                                        close(handle);
                                        stage10 = 1;
                                        break;
                              }
                          }
                         if (yyvsp[-6] != 3)
                          { sprintf(buffer_temp, "%2d%1d%3d%3d%3d", yyvsp[-9], yyvsp[-6],
yyvsp[-2], yyvsp[-1], yyvsp[0]);
                            strcat(buffer10, buffer_temp); }
                         break;
```

```
                case 6: if (yyvsp[-6] == 2)
                          stage10++;
                        else
                          { switch(yyvsp[-6]) {
                              case 3: sprintf(buffer_temp, "223%3d%3d%3d", 0, 0, 0);
                                      strcat(buffer10, buffer_temp);
                                      sprintf(buffer_temp, "%2d%1d%3d%3d%3d\n", yyvsp[-9],
yyvsp[-6], yyvsp[-2], yyvsp[-1], yyvsp[0]);
                                      strcat(buffer10, buffer_temp);
                                      while ((handle =
sopen(append_file_name_to_path("XPLEX10.TXT"), O_CREAT | O_WRONLY, SH_DENYNO,
S_IWRITE)) == FALSE);
                                      length = filelength(handle) + (long)XPLEX10_LENGTH + 1;
                                      while (locking(handle, LK_NBRLCK, length) != -1);
                                      lseek(handle, 0L, SEEK_END);
                                      write(handle, buffer10, XPLEX10_LENGTH);
                                      lseek(handle, 0L, SEEK_SET);
                                      while (locking(handle, LK_UNLCK, length) != -1);
                                      close(handle);
                                      stage10 = 1;
                                      break;
                                  }
                          }
                        if (yyvsp[-6] != 3)
                          { sprintf(buffer_temp, "%2d%1d%3d%3d%3d", yyvsp[-9], yyvsp[-6],
yyvsp[-2], yyvsp[-1], yyvsp[0]);
                            strcat(buffer10, buffer_temp); }
                        break;
                case 7: if (yyvsp[-6] == 3)
                          { switch(yyvsp[-6]) {
                              case 3: sprintf(buffer_temp, "%2d%1d%3d%3d%3d\n", yyvsp[-9],
yyvsp[-6], yyvsp[-2], yyvsp[-1], yyvsp[0]);
                                      strcat(buffer10, buffer_temp);
                                      while ((handle =
sopen(append_file_name_to_path("XPLEX10.TXT"), O_CREAT | O_WRONLY, SH_DENYNO,
S_IWRITE)) == FALSE);
                                      length = filelength(handle) + (long)XPLEX10_LENGTH + 1;
                                      while (locking(handle, LK_NBRLCK, length) != -1);
                                      lseek(handle, 0L, SEEK_END);
                                      write(handle, buffer10, XPLEX10_LENGTH);
                                      lseek(handle, 0L, SEEK_SET);
                                      while (locking(handle, LK_UNLCK, length) != -1);
                                      close(handle);
                                      stage10 = 1;
                                      break;
                                  }
                          }
                        break;
              }
            }
break;
case 12:
#line 228 "testya.y"
{
              if (strcmp(tme, previous_time) == 0)
                count = count + 1;
              else { strcpy(previous_time, tme);
                count = 1; }
              sprintf(buffer, "%s%3d%d%2d%s%s\n", time_stmp, count, yyvsp[-4], yyvsp[-3],
node, dl_string);
```

```
                while ((handle = sopen(append_file_name_to_path("XPLEX04.TXT"), O_CREAT |
O_WRONLY, SH_DENYNO, S_IWRITE)) == FALSE);
                string_length = strlen(file_buffer);
                length = filelength(handle) + (long)string_length + 1;
                while (locking(handle, LK_NBRLCK, length) != -1);
                lseek(handle, 0L, SEEK_END);
                write(handle, file_buffer, string_length);
                lseek(handle, 0L, SEEK_SET);
                while (locking(handle, LK_UNLCK, length) != -1);
                close(handle);
                }
break;
case 13:
#line 247 "testya.y"
{
                if (strcmp(tme, previous_time) == 0)
                  count = count + 1;
                else { strcpy(previous_time, tme);
                  count = 1; }
                sprintf(file_buffer, "%s%3d%d%2d%s\n", time_stmp, count, yyvsp[-4], yyvsp[-3],
node);
                while ((handle = sopen(append_file_name_to_path("XPLEX02.TXT"), O_CREAT |
O_WRONLY, SH_DENYNO, S_IWRITE)) == FALSE);
                length = filelength(handle) + (long)XPLEX02_LENGTH + 1;
                while (locking(handle, LK_NBRLCK, length) != -1);
                lseek(handle, 0L, SEEK_END);
                write(handle, file_buffer, XPLEX02_LENGTH);
                lseek(handle, 0L, SEEK_SET);
                while (locking(handle, LK_UNLCK, length) != -1);
                close(handle);
                }
break;
case 14:
#line 265 "testya.y"
{
                if (strcmp(tme, previous_time)==0)
                  count = count + 1;
                else { strcpy(previous_time, tme);
                  count = 1; }
                sprintf(file_buffer, "%s%3d%d%2d%s%s\n", time_stmp, count, yyvsp[-4], yyvsp[-3],
node, link);
                while ((handle = sopen(append_file_name_to_path("XPLEX03.TXT"), O_CREAT |
O_WRONLY, SH_DENYNO, S_IWRITE)) == FALSE);
                length = filelength(handle) + (long)XPLEX03_LENGTH + 1;
                while (locking(handle, LK_NBRLCK, length) != -1);
                lseek(handle, 0L, SEEK_END);
                write(handle, file_buffer, XPLEX03_LENGTH);
                lseek(handle, 0L, SEEK_SET);
                while (locking(handle, LK_UNLCK, length) != -1);
                close(handle);
                }
break;
case 15:
#line 284 "testya.y"
{
                if (strcmp(tme, previous_time)==0)
                  count = count + 1;
                else { strcpy(previous_time, tme);
                  count = 1; }
                switch (stage11) {
```

```
                case 1: if (yyvsp[-5] == 23)
                          { sprintf(buffer11, "%s%3d%d%s%s%5u%2d%5u%5u", time_stmp, count,
yyvsp[-6], node, link, yyvsp[-3], yyvsp[-5], yyvsp[-2], yyvsp[-1]);
                            stage11++; }
                        break;

                case 2: if (yyvsp[-5] == 24)
                          { sprintf(buffer_temp, "%2d%5u%5u", yyvsp[-5], yyvsp[-2], yyvsp[-1]);
                            strcat(buffer11, buffer_temp);
                            stage11++; }
                        else
                          stage11 = 1;
                        break;
                case 3: if (yyvsp[-5] == 25)
                          { sprintf(buffer_temp, "%2d%5u%5u", yyvsp[-5], yyvsp[-2], yyvsp[-1]);
                            strcat(buffer11, buffer_temp);
                            stage11++; }
                        else
                          stage11 = 1;
                        break;
                case 4: if (yyvsp[-5] == 26)
                          { sprintf(buffer_temp, "%2d%5u%5u", yyvsp[-5], yyvsp[-2], yyvsp[-1]);
                            strcat(buffer11, buffer_temp);
                            stage11++; }
                          else
                            stage11 = 1;
                        break;
                default: stage11 = 1;
                         break;
                }
              }
break;
case 16:
#line 323 "testya.y"
{
              if (strcmp(tme, previous_time)==0)
                count = count + 1;
              else
                { strcpy(previous_time, tme);
                  count = 1; }
              switch (stage11) {
                case 5: if (yyvsp[-5] == 27)
                          { sprintf(buffer_temp, "%2d%3u%3u%3u", yyvsp[-5], yyvsp[-2], yyvsp[-
1], yyvsp[0]);
                            strcat(buffer11, buffer_temp);
                            stage11++; }
                        else
                          stage11 = 1;
                        break;
                case 6: if (yyvsp[-5] == 28)
                          { sprintf(buffer_temp, "%2d%3u%3u%3u", yyvsp[-5], yyvsp[-2], yyvsp[-
1], yyvsp[0]);
                            strcat(buffer11, buffer_temp);
                            stage11++; }
                          else
                            stage11 = 1;
                        break;
                case 7: if (yyvsp[-5] == 29)
                          { sprintf(buffer_temp, "%2d%3u%3u%3u\n", yyvsp[-5], yyvsp[-2],
yyvsp[-1], yyvsp[0]);
```

```
                        strcat(buffer11, buffer_temp);
                        while ((handle = sopen(append_file_name_to_path("XPLEX11.TXT"),
O_CREAT | O_WRONLY, SH_DENYNO, S_IWRITE)) == FALSE);
                        length = filelength(handle) + (long)XPLEX11_LENGTH + 1;
                        while (locking(handle, LK_NBRLCK, length) != -1);
                        lseek(handle, 0L, SEEK_END);
                        write(handle, buffer11, XPLEX11_LENGTH);
                        lseek(handle, 0L, SEEK_SET);
                        while (locking(handle, LK_UNLCK, length) != -1);
                        close(handle);
                        stage11 = 1;
                       }
                      else
                       stage11 = 1;
                      break;
                default: stage11 = 1;
                       break;
                }
               }
break;
case 17:
#line 367 "testya.y"
{
               if (strcmp(tme, previous_time) == 0)
                 count = count + 1;
               else
                 { strcpy(previous_time, tme);
                   count = 1; }
               sprintf(file_buffer, "%s%3d%d%2d%s%s%s\n", time_stmp, count, yyvsp[-4], yyvsp[-
3], node, link, channel);.
               while ((handle = sopen(append_file_name_to_path("XPLEX05.TXT"), O_CREAT |
O_WRONLY, SH_DENYNO, S_IWRITE)) == FALSE);
               length = filelength(handle) + (long)XPLEX05_LENGTH + 1;
               while (locking(handle, LK_NBRLCK, length) != -1);
               lseek(handle, 0L, SEEK_END);
               write(handle, file_buffer, XPLEX05_LENGTH);
               lseek(handle, 0L, SEEK_SET);
               while (locking(handle, LK_UNLCK, length) != -1);
               close(handle);
               }
break;
case 18:
#line 387 "testya.y"
{
               if (strcmp(tme, previous_time) == 0)
                 count = count + 1;
               else
                 { strcpy(previous_time, tme);
                   count = 1; }
               sprintf(file_buffer, "%s%3d%d%2d%s%s%s%5u%5u\n", time_stmp, count, yyvsp[-7],
yyvsp[-6], node, link, channel, yyvsp[-4], yyvsp[-3]);
               while ((handle = sopen(append_file_name_to_path("XPLEX08.TXT"), O_CREAT |
O_WRONLY, SH_DENYNO, S_IWRITE)) == FALSE);
               length = filelength(handle) + (long)XPLEX08_LENGTH + 1;
               while (locking(handle, LK_NBRLCK, length) != -1);
               lseek(handle, 0L, SEEK_END);
               write(handle, file_buffer, XPLEX08_LENGTH);
               lseek(handle, 0L, SEEK_SET);
               while (locking(handle, LK_UNLCK, length) != -1);
               close(handle);
```

```
                }
break;
case 19:
#line 407 "testya.y"
{
                if (strcmp(tme, previous_time) == 0)
                  count = count + 1;
                else
                  { strcpy(previous_time, tme);
                    count = 1; }
                sprintf(file_buffer, "%s%3d%d%2d%s%s%s%5u%5u\n", time_stmp, count, yyvsp[-7],
yyvsp[-6], node, link, channel, yyvsp[-4], yyvsp[-3]);
                while ((handle = sopen(append_file_name_to_path("XPLEX08.TXT"), O_CREAT |
O_WRONLY, SH_DENYNO, S_IWRITE)) == FALSE);
                length = filelength(handle) + (long)XPLEX08_LENGTH + 1;
                while (locking(handle, LK_NBRLCK, length) !=-1);
                lseek(handle, 0L, SEEK_END);
                write(handle, file_buffer, XPLEX08_LENGTH);
                lseek(handle, 0L, SEEK_SET);
                while (locking(handle, LK_UNLCK, length) != -1);
                close(handle);
                }
break;
case 20:
#line 426 "testya.y"
{ if (strcmp(tme, previous_time) == 0)
                    count = count + 1;
                else
                  { strcpy(previous_time, tme);
                    count = 1; }
                sprintf(file_buffer, "%s%3d%d%2d%s%s%s%2d\n", time_stmp, count, yyvsp[-5],
yyvsp[-4], node, link, channel, yyvsp[-2]);
                while ((handle = sopen(append_file_name_to_path("XPLEX07.TXT"), O_CREAT |
O_WRONLY, SH_DENYNO, S_IWRITE)) == FALSE);
                length = filelength(handle) + (long)XPLEX07_LENGTH + 1;
                while (locking(handle, LK_NBRLCK, length) != -1);
                lseek(handle, 0L, SEEK_END);
                write(handle, file_buffer, XPLEX07_LENGTH);
                lseek(handle, 0L, SEEK_SET);
                while (locking(handle, LK_UNLCK, length) != -1);
                close(handle);
                }
break;
case 21:
#line 444 "testya.y"
{ if (strcmp(tme, previous_time) == 0)
                    count = count + 1;
                else
                  { strcpy(previous_time, tme);
                    count = 1; }
                sprintf(file_buffer, "%s%3d%d%2d%s%s%s%s%s%s\n", time_stmp, count, yyvsp[-4],
yyvsp[-3], orig_node, orig_link, orig_channel, node, link, channel);
                while ((handle = sopen(append_file_name_to_path("XPLEX12.TXT"), O_CREAT |
O_WRONLY, SH_DENYNO, S_IWRITE)) == FALSE);
                length = filelength(handle) + (long)XPLEX12_LENGTH + 1;
                while (locking(handle, LK_NBRLCK, length) != -1);
                lseek(handle, 0L, SEEK_END);
                write(handle, file_buffer, XPLEX12_LENGTH);
                lseek(handle, 0L, SEEK_SET);
                while (locking(handle, LK_UNLCK, length) != -1);
```

```
                close(handle);
                }
break;
case 22:
#line 462 "testya.y"
{
                if (strcmp(tme, previous_time) == 0)
                  count = count + 1;
                else
                  { strcpy(previous_time, tme);
                    count = 1; }
                sprintf(file_buffer, "%s%3d%d%2d%s%s%s%s\n", time_stmp, count, yyvsp[-5],
yyvsp[-4], node, link, channel, dl_string);
                while ((handle = sopen(append_file_name_to_path("XPLEX06.TXT"), O_CREAT |
O_WRONLY, SH_DENYNO, S_IWRITE)) == FALSE);
                string_length = strlen(file_buffer);
                length = filelength(handle) + (long)string_length + 1;
                while (locking(handle, LK_NBRLCK, length) != -1);
                lseek(handle, 0L, SEEK_END);
                write(handle, file_buffer, string_length);
                lseek(handle, 0L, SEEK_SET);
                while (locking(handle, LK_UNLCK, length) != -1);
                close(handle);
                }
break;
case 23:
#line 482 "testya.y"
{
                if (strcmp(tme, previous_time) == 0)
                  count = count + 1;
                else { strcpy(previous_time, tme);
                  count = 1; }
                sprintf(file_buffer, "%s%3d%d%2d%s%s%s%s\n", time_stmp, count, yyvsp[-4],
yyvsp[-3], node, link, channel, dl_string);
                while ((handle = sopen(append_file_name_to_path("XPLEX06.TXT"), O_CREAT |
O_WRONLY, SH_DENYNO, S_IWRITE)) == FALSE);
                string_length = strlen(file_buffer);
                length = filelength(handle) + (long)string_length + 1;
                while (locking(handle, LK_NBRLCK, length) != -1);
                lseek(handle, 0L, SEEK_END);
                write(handle, file_buffer, string_length);
                lseek(handle, 0L, SEEK_SET);
                while (locking(handle, LK_UNLCK, length) != -1);
                close(handle);
                }
break;
case 24:
#line 500 "testya.y"
{ yyval = yyvsp[-1];}
break;
case 25:
#line 503 "testya.y"
{ yyval = yyvsp[-1];}
break;
case 26:
#line 506 "testya.y"
{ yyval = yyvsp[-1];}
break;
case 27:
#line 509 "testya.y"
```

```
{ yyval = yyvsp[-1];}
break;
case 28:
#line 512 "testya.y"
{ yyval = yyvsp[-1];}
break;
case 29:
#line 515 "testya.y"
{ yyval = yyvsp[0];}
break;
case 30:
#line 518 "testya.y"
{ yyval = yyvsp[0]; }
break;
case 31:
#line 519 "testya.y"
{ yyval = yyvsp[0]; }
break;
case 32:
#line 520 "testya.y"
{ yyval = yyvsp[0]; }
break;
case 35:
#line 529 "testya.y"
{
            switch(yyvsp[0]) {
              case 331: yyval = 1;
                        break;
              case 440: yyval = 2;
                        break;
              case 445: yyval = 3;
                        break;
              case 330: yyval = 4;
                        break;
              }
            }
break;
#line 939 "y_tab.c"
    }
    yyssp -= yym;
    yystate = *yyssp;
    yyvsp -= yym;
    yym = yylhs[yyn];
    if (yystate == 0 && yym == 0)
    {
#if YYDEBUG
        if (yydebug)
            printf("yydebug: after reduction, shifting from state 0 to\
 state %d\n", YYFINAL);
#endif
        yystate = YYFINAL;
        *++yyssp = YYFINAL;
        *++yyvsp = yyval;
        if (yychar < 0)
        {
            if ((yychar = yylex()) < 0) yychar = 0;
#if YYDEBUG
            if (yydebug)
            {
                yys = 0;
```

```
                if (yychar <= YYMAXTOKEN) yys = yyname[yychar];
                if (!yys) yys = "illegal-symbol";
                printf("yydebug: state %d, reading %d (%s)\n",
                        YYFINAL, yychar, yys);
            }
#endif
        }
        if (yychar == 0) goto yyaccept;
        goto yyloop;
    }
    if ((yyn = yygindex[yym]) && (yyn += yystate) >= 0 &&
            yyn <= YYTABLESIZE && yycheck[yyn] == yystate)
        yystate = yytable[yyn];
    else
        yystate = yydgoto[yym];
#if YYDEBUG
    if (yydebug)
        printf("yydebug: after reduction, shifting from state %d \
to state %d\n", *yyssp, yystate);
#endif
    if (yyssp >= yyss + yystacksize - 1)
    {
        goto yyoverflow;
    }
    *++yyssp = yystate;
    *++yyvsp = yyval;
    goto yyloop;
yyoverflow:
    yyerror("yacc stack overflow");
yyabort:
    return (1);
yyaccept:
    return (0);
}
```

```
%{

char pop()
{
        if (input_stackptr.top == 0)
          printf("Trying to pop a character off the stack which doesn't exist\n");
        else
          return(input_stackptr.inputChar[-- input_stackptr.top]);
}

push(c)
        char c;
{
        if (input_stackptr.top == 256)
          printf("Stack overflow in unput()\n");
        else
          input_stackptr.inputChar[input_stackptr.top ++] = c;
}

int empty()
{
        if (input_stackptr.top == 0)
          return(TRUE);
        else
          return(FALSE);
}

# undef input
input()
{
```

```
        char return_value;

        if (empty())
         {
           return_value = read_com();
            return(return_value);
         }
        else
        {
          return_value = pop();
         return(return_value);
        }
}

# undef unput
unput(c)
{
        push(c);
}

%}

WS      [ \t]

%%

"\r"                    ;

"\n"                            { printf("%s", yytext);
                        return(NEW_LINE); }
```

```
{WS}+                   { printf("%s", yytext); }

"minute(s)"             { printf("%s", yytext);
                          return(MIN_COMMENT); }

"Tx"                    { printf("%s", yytext);
                          return(TRANS_COMMENT); }

"Rx"                    { printf("%s", yytext);
                          return(REC_COMMENT); }

"I Frames"              { printf("%s", yytext);
                          return(COMMENT); }

"S Frames"              { printf("%s", yytext);
                          return(COMMENT); }

"E Frames"              { printf("%s", yytext);
                          return(COMMENT); }

"Link Tx"               { printf("%s", yytext);
                          return(UTIL_COMMENT); }

"Rx Utilization %"      { printf("%s", yytext);
                          return(UTIL_COMMENT); }

"Buffer"                { printf("%s", yytext);
                          return(BUFFER_COMMENT); }
```

```
"0- 7"          { printf("%s", yytext);
                  return(CHAN8); }

"8-15"          { printf("%s", yytext);
                  return(CHAN16); }

"16-23"         { printf("%s", yytext);
                  return(CHAN24); }

"24-31"         { printf("%s", yytext);
                  return(CHAN32); }

[1-9][0-9]*|[0] { printf("%s", yytext);
                  yylval = atoi(yytext);
                        return(NUMB); }

"*****"         { printf("%s", yytext);
                  yylval = 65536;
                  return(NUMB); }

"(Node "        { printf("%s", yytext);
                  return(NODE); }

"Channel"       { printf("%s", yytext);
                  return(CHANNEL_COMMENT); }


"(x100)"              { printf("%s", yytext);
                  return(TIMES_ONE_HUNDRED); }
```

```
"Characters"            { printf("%s", yytext);
                          return(CHARACTERS_COMMENT); }

"(x1000)"               { printf("%s", yytext);
                          return(TIMES_ONE_THOUSAND); }

"channel"               { printf("%s", yytext);
                          return(END_CHANNEL_COMMENT); }

"Destination Busy At"          { printf("%s", yytext);
                         return(DEST_BUSY); }

"***** Data Link Timeout" { printf("%s", yytext);
                           return(DATA_LINK_TIMEOUT); }

"***** Invalid response"  { printf("%s", yytext);
                          return(INVALID_RESPONSE); }

"***** Link not a dataplexer"          { printf("%s", yytext);
                                  return(LINK_NOT_DATAPLEXER); }
"Not Properly"                  { printf("%s", yytext);
                         return(NOT_PROPERLY_CONFIGURED); }
"***** Messages Pending" { printf("%s", yytext);
                          return(MESSAGES_PENDING); }
"STANDBY"                       { printf("%s", yytext);
                            return(STANDBY); }
%%
```

```
%{

# include <h:\users\lbw\c600\include\stdio.h>
# include <h:\users\lbw\c600\include\string.h>

# define TRUE 1
# define FALSE 0
# define DBUFFER_LENGTH 28
# define DTRANS_LENGTH 179
# define DRECEIVE_LENGTH 179
# define S_IWRITE 0000200 /* write permission, owner */

extern char xplexer_dataplexer[7];
extern char *datetime;
char path_name[80];
int loop;
int handle;
long int length;
int count;
char *buffer_trans;
char *buffer_receive;
char buffer[DBUFFER_LENGTH];
char buffer_temp_trans[59];
char buffer_temp_receive[59];

FILE *fp;

struct input_stack { char inputChar[256];
                     int top;
                   } input_stackptr;

%}

%token NUMB NEW_LINE COMMENT TRANS_COMMENT REC_COMMENT
BUFFER_COMMENT
%token UTIL_COMMENT CHAN8 CHAN16 CHAN24 CHAN32 NODE MIN_COMMENT
CHANNEL_COMMENT
%token TIMES_ONE_HUNDRED CHARACTERS_COMMENT TIMES_ONE_THOUSAND
DEST_BUSY
%token DATA_LINK_TIMEOUT INVALID_RESPONSE LINK_NOT_DATAPLEXER
%token END_CHANNEL_COMMENT NOT_PROPERLY_CONFIGURED
MESSAGES_PENDING STANDBY

%start dataplexer


%%

dataplexer     : /* empty */

       |       NUMB dataplexer

       |       NUMB MIN_COMMENT dataplexer

       |       NEW_LINE dataplexer

       |       MESSAGES_PENDING dataplexer

       |       STANDBY { yyerrok; YYACCEPT; }
```

```
        |       DEST_BUSY { yyerrok; YYACCEPT; }

        |       DATA_LINK_TIMEOUT { yyerrok; YYACCEPT; }

        |       INVALID_RESPONSE { yyerrok; YYACCEPT; }

        |       LINK_NOT_DATAPLEXER { yyerrok; YYACCEPT; }

        |       NOT_PROPERLY_CONFIGURED { yyerrok; YYACCEPT; }

        |       END_CHANNEL_COMMENT NUMB { yyerrok; YYACCEPT; }

        |       BUFFER_COMMENT NUMB NUMB NUMB

                {

                        /* return from the parser in order to continue
                           interrogating the network for more data or
                           to wait for the next interrogation period. */

                sprintf(buffer, "%s%s%3d%3d%3d\n", datetime, xplexer_dataplexer, $2, $3, $4);
                while((handle = sopen(append_file_name_to_path("DBUFFER.TXT"), O_CREAT |
O_WRONLY, SH_DENYNO, S_IWRITE)) == FALSE);
                length = filelength(handle) + (long)DBUFFER_LENGTH + 1;
                while(locking(handle, LK_NBRLCK, length) != -1);
                lseek(handle, 0L, SEEK_END);
                write(handle, buffer, DBUFFER_LENGTH);
                lseek(handle, 0L, SEEK_SET);
                while(locking(handle, LK_UNLCK, length) != -1);
                close(handle);

                sprintf(buffer_temp_receive, "%5u%5u%5u%5u%5u%5u%5u%5u", 0, 0, 0, 0, 0, 0, 0,
0);
                sprintf(buffer_temp_trans, "%5u%5u%5u%5u%5u%5u%5u%5u", 0, 0, 0, 0, 0, 0, 0,
0);
                    switch (count) {
                      case 2: for(loop = 1; loop <= 2; loop++)
                                { strcat(buffer_receive, buffer_temp_receive);
                                  strcat(buffer_trans, buffer_temp_trans); }
                                sprintf(buffer_temp_receive, "%5u%5u%5u%5u%5u%5u%5u%5u\n",
0, 0, 0, 0, 0, 0, 0, 0);
                                sprintf(buffer_temp_trans, "%5u%5u%5u%5u%5u%5u%5u%5u\n", 0,
0, 0, 0, 0, 0, 0, 0);
                              break;
                      case 3:  strcat(buffer_receive, buffer_temp_receive);
                               strcat(buffer_trans, buffer_temp_trans);
                               sprintf(buffer_temp_receive, "%5u%5u%5u%5u%5u%5u%5u%5u\n",
0, 0, 0, 0, 0, 0, 0, 0);
                               sprintf(buffer_temp_trans, "%5u%5u%5u%5u%5u%5u%5u%5u\n", 0,
0, 0, 0, 0, 0, 0, 0);
                               break;
                      case 4: sprintf(buffer_temp_receive, "%5u%5u%5u%5u%5u%5u%5u%5u\n", 0,
0, 0, 0, 0, 0, 0, 0);
                              sprintf(buffer_temp_trans, "%5u%5u%5u%5u%5u%5u%5u%5u\n", 0, 0,
0, 0, 0, 0, 0, 0);
                              break;
                      case 5: break;
                      default: count = 1;
                               break;
                      }
```

```
                    strcat(buffer_receive, buffer_temp_receive);
                    strcat(buffer_trans, buffer_temp_trans);
                    while((handle = sopen(append_file_name_to_path("DRECEIVE.TXT"),
O_CREAT | O_WRONLY, SH_DENYNO, S_IWRITE)) == FALSE);
                    length = filelength(handle) + (long)DRECEIVE_LENGTH + 1;
                    while(locking(handle, LK_NBRLCK, length) != -1);
                    lseek(handle, 0L, SEEK_END);
                    write(handle, buffer_receive, DRECEIVE_LENGTH);
                    lseek(handle, 0L, SEEK_SET);
                    while(locking(handle, LK_UNLCK, length) != -1);
                    close(handle);

                while((handle = sopen(append_file_name_to_path("DTRANS.TXT"), O_CREAT |
O_WRONLY, SH_DENYNO, S_IWRITE)) == FALSE);
                    length = filelength(handle) + (long)DTRANS_LENGTH + 1;
                    while(locking(handle, LK_NBRLCK, length) != -1);
                    lseek(handle, 0L, SEEK_END);
                    write(handle, buffer_trans, DTRANS_LENGTH);
                    lseek(handle, 0L, SEEK_SET);
                    while(locking(handle, LK_UNLCK, length) != -1);
                    close(handle);
                count = 1;
                YYACCEPT;
                }

        |       error { yyerrok; YYACCEPT; }

        |       CHARACTERS_COMMENT TIMES_ONE_THOUSAND NUMB NUMB
NEW_LINE dataplexer

        |       TRANS_COMMENT REC_COMMENT dataplexer

        |       COMMENT TIMES_ONE_HUNDRED NUMB NUMB dataplexer

        |       COMMENT NUMB NUMB dataplexer

        |       COMMENT NUMB NEW_LINE dataplexer

        |       UTIL_COMMENT NUMB NUMB NUMB dataplexer

        |       NODE NUMB dataplexer

        |       CHANNEL_COMMENT TIMES_ONE_HUNDRED dataplexer

        |       TRANS_COMMENT NUMB NUMB NUMB NUMB NUMB NUMB NUMB
NUMB

                {
                switch (count) {
                  case 1: sprintf(buffer_trans, "%s%s%5u%5u%5u%5u%5u%5u%5u%5u", datetime,
xplexer_dataplexer, $2, $4, $5, $6, $7, $8, $9);
                          count++;
                          break;
                  case 4: sprintf(buffer_temp_trans, "%5u%5u%5u%5u%5u%5u%5u%5u\n", $2, $3,
$4, $5, $6, $7, $8, $9);
                          strcat(buffer_trans, buffer_temp_trans);
                          count++;
                          break;
                  default: sprintf(buffer_temp_trans, "%5u%5u%5u%5u%5u%5u%5u%5u", $2, $3,
$4, $5, $6, $7, $8, $9);
```

```
                    strcat(buffer_trans, buffer_temp_trans);
                    count++;
                    break;
                }
            } dataplexer

      |     CHAN8 REC_COMMENT NUMB NUMB NUMB NUMB NUMB NUMB NUMB
NUMB

            {
                sprintf(buffer_receive, "%s%s%5u%5u%5u%5u%5u%5u%5u%5u", datetime,
xplexer_dataplexer, $3, $4, $5, $6, $7, $8, $9, $10);
            } dataplexer

      |     CHAN16 REC_COMMENT NUMB NUMB NUMB NUMB NUMB NUMB NUMB
NUMB

            {
              sprintf(buffer_temp_receive, "%5u%5u%5u%5u%5u%5u%5u%5u", $3, $4, $5, $6,
$7, $8, $9, $10);
              strcat(buffer_receive, buffer_temp_receive);
            } dataplexer

      |     CHAN24 REC_COMMENT NUMB NUMB NUMB NUMB NUMB NUMB NUMB
NUMB

            {
              sprintf(buffer_temp_receive, "%5u%5u%5u%5u%5u%5u%5u%5u", $3, $4, $5, $6,
$7, $8, $9, $10);
              strcat(buffer_receive, buffer_temp_receive);
            } dataplexer

      |     CHAN32 REC_COMMENT NUMB NUMB NUMB NUMB NUMB NUMB NUMB
NUMB

            {
              sprintf(buffer_temp_receive, "%5u%5u%5u%5u%5u%5u%5u%5u\n", $3, $4, $5,
$6, $7, $8, $9, $10);
              strcat(buffer_receive, buffer_temp_receive);
            } dataplexer


%%

# include "LEXYY.C"

yyerror(s)
  char *s;
{
  fprintf(stderr, "Error: %s\n", s);
}


yywrap()
{
  return 1;
}

char *
append_file_name_to_path(file_name)
```

```
  char file_name[12];
{
  char new_path[268];
  strcpy(new_path, path_name);
  strcat(new_path, file_name);
  return(new_path);
}
```

```
# include <h:\users\lbw\c600\include\stdio.h>
# include <h:\users\lbw\c600\include\string.h>
# include <h:\users\lbw\c600\include\io.h>
# include <h:\users\lbw\c600\include\sys\locking.h>
# include <h:\users\lbw\c600\include\fcntl.h>
# include <h:\users\lbw\c600\include\process.h>
# include <h:\users\lbw\c600\include\share.h>
# include <h:\users\lbw\c600\include\malloc.h>
# include "Y_TAB.C"

# define S_IREAD        0000400         /* read permission, owner */
# define DAY 2400
# define HOURS_IN_DAY 1440
# define MINS_IN_HOUR 60
# define HOUR 100
# define TRUE 1
# define FALSE 0
# define NOTFINISHED 1  /* never finish i.e. endless loop */
# define CR 13
# define LF 10
# define MAXINPUTLENGTH 256

extern char path_name[80];
int new_time, oldtime;
int interrogate;
extern unsigned int fifo_size, fifo_index, read_index;
extern int count;
char xplexer_dataplexer[7];
int interrogation_period, time_period;
int speed, port_number, word_length;
char batch_name[MAXINPUTLENGTH];
int loop, outer_loop;

FILE *fp;
int handle, status;
long length;
char *datetime;
int remainder, quotient, total_hours, old_total_hours;

typedef struct batch_node {
    char *name;
    struct batch_node *next;
    } batch_node, *batch_nodeptr;

batch_nodeptr batch_list = NULL;
batch_nodeptr batch_listptr = NULL;
int num_batchfiles;
batch_nodeptr newnode;

void interrogate_node(batch_file)
     char batch_file[13];

{
char ch;
  fifo_size = 0;
  fifo_index = 0;
  read_index = 0;   /* flush the input buffer by resetting the
                       size and the index of the array. This
                       is done before the interrogation so that
                       anything left lying around in the buffer
```

```
                    from the previous interrogation is removed. */

/* interrogate the dataplexer for channel_throughput and
    buffer utilization statistics */

  fp = fopen(batch_file, "r");
  fscanf(fp, "%s\n", xplexer_dataplexer);

  printf("\n\nCURRENTLY INTERROGATING XPLEXER/DATAPLEXER  %c%c%c/%s\n\n",
xplexer_dataplexer[0],
                        xplexer_dataplexer[1], xplexer_dataplexer[2],
                        xplexer_dataplexer + 3);
  printf("CURRENT BATCH FILE = %s\n\n", batch_file);

  while ((ch = fgetc(fp)) != EOF)
  {
  printf("%c", ch);

/* Strip out the LINE FEED character to the Network, it only takes
   CARRIAGE RETURN characters                                      */

  if (ch == LF) {
    write_to_com(CR);


/* Create a delay between lines sent otherwise the Tellabs Network
   is swamped with the incoming data                               */

    for(outer_loop = 0; outer_loop <= 15; outer_loop++)
    for(loop = 0; loop <= 32000; loop++);
  }
  else
    write_to_com(ch);
  }
  fclose(fp);

/* call the parser to pull out the important information from the interrogation
   and then return in order to interrogate the next dataplexer. The buffer
   size for information from an enquiry is set at 32kbytes for the information
   from the COM port. This is set in COMIO.C. */

  yyparse();

/* Reset the COM PORT buffer */

  memset(input_stackptr.inputChar, '\0', 256);
  input_stackptr.top = 0;
  fifo_index = 0;
  fifo_size = 0;
  read_index = 0;

}

/* ------------------------------------------------------------------------ */

main()
{
  count = 1;
  while ((datetime = (char *)malloc(13)) == NULL );
  while ((buffer_trans = (char *)malloc(DTRANS_LENGTH)) == NULL );
```

```
  while ((buffer_receive = (char *)malloc(DRECEIVE_LENGTH)) == NULL );

/* Initialize the COM PORT buffer */

  memset(input_stackptr.inputChar, '\0', 256);
  input_stackptr.top = 0;
  fifo_size = 0;
  read_index = 0;
  interrogation_period = 0;
  fp = fopen("COMMDCFG.DAT", "r");
  fscanf(fp, "%d%d%d%s%d", &speed, &port_number, &word_length, path_name,
&interrogation_period);
  time_period = interrogation_period;
  fscanf(fp, "%s", batch_name);
  newnode = (batch_nodeptr) malloc(sizeof(batch_node));
  if (newnode == NULL)
    printf("Not Enough Memory to Hold all Batch Files\n");
  newnode->name = strdup(batch_name);
  newnode->next = batch_list;
  batch_list = newnode;
  num_batchfiles++;
  batch_listptr = batch_list;

  while (fscanf(fp, "%s", batch_name) != EOF) {
    newnode = (batch_nodeptr) malloc(sizeof(batch_node));
    if (newnode == NULL)
      printf("Not Enough Memory to Hold all Batch Files\n");
    newnode->name = strdup(batch_name);
    newnode->next = NULL;
    batch_listptr->next = newnode;
    batch_listptr = newnode;
    num_batchfiles++;
    }
  batch_listptr = batch_list;
  fclose(fp);

  open_com(speed, port_number, word_length);

  while ((handle = sopen("DATETIME.TXT", O_CREAT | O_RDONLY, SH_DENYNO,
S_IREAD)) == FALSE);
  length = 12;
  while ((status = locking(handle, LK_NBRLCK, length)) != 0);

  /* critical section of code. */

  lseek(handle, 0L, SEEK_SET);
  read(handle, datetime, 12);

  /* end of critical section of code. */

  lseek(handle, 0L, SEEK_SET);
  while ((status = locking(handle, LK_UNLCK, length)) != 0);

  close(handle);

  oldtime = atoi(datetime + 8); /* time now equals the time field out of the
                                   date/time file. i.e. it removes the date
                                   field. */

  remainder = oldtime % HOUR;
```

```
  quotient = oldtime / HOUR;

  old_total_hours = (quotient * MINS_IN_HOUR) + remainder;

  interrogate = FALSE;

  while (NOTFINISHED)
   {

    while (interrogate)
     {
       while ((handle = sopen("DATETIME.TXT", O_CREAT | O_RDONLY, SH_DENYNO,
S_IREAD)) == FALSE);
       length = filelength(handle);
       while ((status = locking(handle, LK_NBRLCK, length)) != 0);

       /* critical section of code. */

       read(handle, datetime, 12);

       /* end of critical section of code. */

       lseek(handle, 0L, SEEK_SET);
       while ((status = locking(handle, LK_UNLCK, length)) != 0);

       close(handle);

       new_time = atoi(datetime + 8);
       remainder = new_time % HOUR;
       quotient = new_time / HOUR;
       total_hours = (quotient * MINS_IN_HOUR) + remainder;

       if (total_hours >= old_total_hours)
         {
         if (total_hours >= (old_total_hours + time_period))
           { oldtime = new_time;
             old_total_hours = total_hours;
             interrogate = FALSE;
           }
         }
       else
         {
         if (total_hours >= ((old_total_hours + time_period) % HOURS_IN_DAY))
           { oldtime = new_time;
             old_total_hours = total_hours;
             interrogate = FALSE;
           }
         }

       }

  printf("STARTING INTERROGATION\n\n");

/* calls a function which takes as its parameter, the name of the batch
   file which contains the user key commands necessary to interrogate
   the required dataplexer. */

  while (batch_listptr != NULL) {
    interrogate_node(batch_listptr->name);
    batch_listptr = batch_listptr->next;
```

```
    }
  batch_listptr = batch_list;
  printf("\n\nFINISHED INTERROGATION\n\n");

  interrogate = TRUE;
  }
  free(datetime);
  close_com();
}
```

2 . 2 8
..\TXTFILES\
180
DATA003.BAT
DATA004.BAT
DATA005.BAT
DATA100.BAT.
DATA101.BAT
DATA102.BAT
DATA107.BAT
DATA108.BAT
DATA109.BAT
DATA202.BAT
DATA203.BAT

000003
COMMAND0
▒▒MODE
n
2
3
3
2
1
R
E
E
E

000004
COMMAND0
▒▒MODE
n
2
4
3
2
1
R
E
E
E

000005
COMMAND0
▒▒MODE
n
2
5
3
2
1
R
E
E
E

001000
COMMAND1
▒▒MODE
n
2
0
3
2
1
R
E
E
E

001001
COMMAND1
▒▒MODE
n
2
1
3
2
1
R
E
E
E

001008
COMMAND1
▒▒MODE
n
2
8
3
2
1
R
E
E
E

001009
COMMAND1
▒▒MODE
n
2
9
3
2
1
R
E
E
E

002002
COMMAND2
▒▒MODE
n
2
2
3
2
1
R
E
E
E

002003
COMMAND2
▒▒MODE
n
2
3
3
2
1
R
E
E
E

/* This is an example of both the COMMDCFG.DAT configuration file and the structure of an interrogation batch file. Note all comments which can be excluded from the original configuration and batch files are enclosed in "/* */".

COMMDCFG.DAT
============

*/

2 2 8

/* the COM port for the command port process

ARGUMENTS : speed - an integer value to indicate baudrate for the com ports.
0 = 300 baud. - allowable but not
1 = 600 baud. recommended.
2 = 1200 baud. - highly recommended.
3 = 2400 baud. - the rest are not
4 = 4800 baud. recommended for
5 = 9600 baud. during peak loads.
6 = 19200 baud.
7 = 38400 baud.

port_number - 1 = COM1. - default.
2 = COM2.

word_length - 7 = 7 bits.
8 = 8 bits. - setup for Tellabs.
*/

D:\TELLABS\TXTFILES\

/*
the destination directory for the database text files

ARGUMENTS : drive: \directory path\
*/

7

/*
the number of minutes between interrogations
*/

DATA003.BAT
DATA004.BAT
DATA005.BAT
DATA100.BAT
DATA101.BAT
DATA102.BAT
DATA107.BAT
DATA108.BAT
DATA109.BAT
DATA202.BAT
DATA203.BAT

```
        /*
                the list of interrogation batch file names which CMDPORT.EXE
                will read from.


Note: the fields for the COM port are tab delimited (or space) and the path
      name is on the next line. The path name must also exist before trying to
      write to the directory or else it will fail.

-------------------------------------------------------------------------------

DATAnnn.BAT
==========

The actual name of the file can be anything descriptive of the dataplexer to
be interrogated.

*/


000003

        /*
                the Xplexer (first 3 numbers) and Dataplexer (second 3 numbers)
                about to be interrogated. This is used for information to be
                written in the database files and not part of the interrogation
                process.
        */

COMMAND0
▒▒MODE

        /*
                COMMAND0 = log on to node 0 of the network (i.e. COMMAND1 for
                node 1 and COMMAND2 for node 2).
                MODE = password for security reasons.
                        Note: the two control-H characters are backspaces needed
                              for the network as input, otherwise it takes the
                              two characters previously there (i.e. carriage
                              return-line feed) as part of the password.
        */

n

        /*
                NO to menus
        */

2

        /*
                REMOTE option from the Xplexer
        */

3

        /*
                link 3 from the Xplexer to get to the Dataplexer on that link.
```

```
        There can be up to 10 links off an Xplexer to choose from,
        depending if there is a Dataplexer attached to it or not.
    */

3

    /*
        choose the STATISTICS option from the Function menu in TELLABS
    */

2

    /*
        Data Link Statistics
    */

1

    /*
        System Statistics
    */

R

    /*  Reset the statistics after each interrogation
    */

E
E
E

    /*
        this sequence of EXIT commands should take the interrogation
        process back to the login stage
    */
```

```
# include "stdio.h"
# define U(x) x
# define NLSTATE yyprevious=YYNEWLINE
# define BEGIN yybgin = yysvec + 1 +
# define INITIAL 0
# define YYLERR yysvec
# define YYSTATE (yyestate-yysvec-1)
# define YYOPTIM 1
# define YYLMAX BUFSIZ
# define output(c) putc(c,yyout)
# define input() (((yytchar=yysptr>yysbuf?U(*--
yysptr):getc(yyin))==10?(yylineno++,yytchar):yytchar)==EOF?0:yytchar)
# define unput(c) {yytchar= (c);if(yytchar=='\n')yylineno--;*yysptr++=yytchar;}
# define yymore() (yymorfg=1)
# define ECHO fprintf(yyout, "%s",yytext)
# define REJECT { nstr = yyreject(); goto yyfussy;}
int yyleng; extern char yytext[];
int yymorfg;
extern char *yysptr, yysbuf[];
int yytchar;
FILE *yyin = {stdin}, *yyout = {stdout};
extern int yylineno;
struct yysvf {
        struct yywork *yystoff;
        struct yysvf *yyother;
        int *yystops;};
struct yysvf *yyestate;
extern struct yysvf yysvec[], *yybgin;



char pop()

{

        if (input_stackptr.top == 0)

          printf("Trying to pop a character off the stack which doesn't exist\n");

        else

          return(input_stackptr.inputChar[-- input_stackptr.top]);

}



push(c)

        char c;

{

        if (input_stackptr.top == 256)

          printf("Stack overflow in unput()\n");

        else

          input_stackptr.inputChar[input_stackptr.top ++] = c;
```

```
}

int empty()
{
        if (input_stackptr.top == 0)
          return(TRUE);
        else
          return(FALSE);
}

# undef input
input()
{
         char return_value;

        if (empty())
          {
            return_value = read_com();
             return(return_value);
          }
        else
         {
           return_value = pop();
          return(return_value);
         }
}

# undef unput
unput(c)
{
```

```
        push(c);
}


# define YYNEWLINE 10
yylex(){
int nstr; extern int yyprevious;
while((nstr = yylook()) >= 0)
yyfussy: switch(nstr){
case 0:
if(yywrap()) return(0); break;
case 1:
                ;
break;
case 2:
                { printf("%s", yytext);
                        return(NEW_LINE); }
break;
case 3:
                { printf("%s", yytext); }
break;
case 4:
                { printf("%s", yytext);
                        return(MIN_COMMENT); }
break;
case 5:
                        { printf("%s", yytext);
                        return(TRANS_COMMENT); }
break;
case 6:
                        { printf("%s", yytext);
                        return(REC_COMMENT); }
break;
case 7:
                { printf("%s", yytext);
                        return(COMMENT); }
break;
case 8:
                { printf("%s", yytext);
                        return(COMMENT); }
break;
case 9:
                { printf("%s", yytext);
                        return(COMMENT); }
break;
case 10:
                { printf("%s", yytext);
                        return(UTIL_COMMENT); }
break;
```

```
case 11:
       { printf("%s", yytext);
                        return(UTIL_COMMENT); }
break;
case 12:
              { printf("%s", yytext);
                        return(BUFFER_COMMENT); }
break;
case 13:
                      { printf("%s", yytext);
                        return(CHAN8); }
break;
case 14:
              { printf("%s", yytext);
                        return(CHAN16); }
break;
case 15:
              { printf("%s", yytext);
                        return(CHAN24); }
break;
case 16:
              { printf("%s", yytext);
                        return(CHAN32); }
break;
case 17:
      { printf("%s", yytext);
                        yylval = atoi(yytext);
                      return(NUMB); }
break;
case 18:
             { printf("%s", yytext);
                        yylval = 65536;
                        return(NUMB); }
break;
case 19:
            { printf("%s", yytext);
                        return(NODE); }
break;
case 20:
           { printf("%s", yytext);
                        return(CHANNEL_COMMENT); }
break;
case 21:
              { printf("%s", yytext);
                        return(TIMES_ONE_HUNDRED); }
break;
case 22:
```

```
        { printf("%s", yytext);
                        return(CHARACTERS_COMMENT); }
break;
case 23:
              { printf("%s", yytext);
                        return(TIMES_ONE_THOUSAND); }
break;
case 24:
            { printf("%s", yytext);
                        return(END_CHANNEL_COMMENT); }
break;
case 25:
       { printf("%s", yytext);
                        return(DEST_BUSY); }
break;
case 26:
{ printf("%s", yytext);
                         return(DATA_LINK_TIMEOUT); }
break;
case 27:
 { printf("%s", yytext);
                         return(INVALID_RESPONSE); }
break;
case 28:
       { printf("%s", yytext);
                              return(LINK_NOT_DATAPLEXER); }
break;
case 29:
       { printf("%s", yytext);
                        return(NOT_PROPERLY_CONFIGURED); }
break;
case 30:
{ printf("%s", yytext);
                         return(MESSAGES_PENDING); }
break;
case 31:
          { printf("%s", yytext);
                    return(STANDBY); }
break;
case -1:
break;
default:
fprintf(yyout,"bad switch yylook %d",nstr);
} return(0); }
/* end of yylex */
```

```
int yyvstop[] = {
0,

3,
0,

2,
0,

1,
0,

17,
0,

17,
0,

17,
0,

17,
0,

17,
0,

17,
0,

17,
0,

6,
0,

5,
0,

13,
0,

14,
0,

18,
0,

15,
0,
```

16,
0,

19,
0,

21,
0,

12,
0,

23,
0,

20,
0,

10,
0,

31,
0,

24,
0,

9,
0,

7,
0,

8,
0,

4,
0,

22,
0,

29,
0,

11,
0,

25,
0,

27,
0,

30,
0,

26,
0,

```
28,
0,
0};
# define YYTYPE int
struct yywork { YYTYPE verify, advance; } yycrank[] = {
0,0,    0,0,    0,0,    0,0,
0,0,    0,0,    0,0,    0,0,
0,0,    0,0,    1,3,    1,4,
3,3,    0,0,    1,5,    0,0,
0,0,    0,0,    0,0,    0,0,
0,0,    0,0,    0,0,    0,0,
0,0,    0,0,    0,0,    0,0,
0,0,    0,0,    0,0,    0,0,
0,0,    1,3,    16,35,  3,3,
17,36,  28,48,  39,59,  58,78,
0,0,    1,6,    0,0,    1,7,
7,27,   21,40,  8,28,   27,47,
47,66,  1,8,    1,9,    1,10,
1,11,   1,11,   1,11,   1,11,
1,11,   1,12,   1,11,   10,30,
11,11,  10,11,  12,31,  26,46,
29,49,  31,51,  30,50,  1,13,
1,14,   1,15,   1,16,   12,11,
35,55,  29,11,  1,17,   30,11,
36,56,  1,18,   40,60,  1,19,
6,25,   41,61,  46,65,  1,20,
1,21,   1,22,   9,11,   9,11,
9,11,   9,11,   9,11,   9,11,
9,29,   9,11,   9,11,   9,11,
48,67,  21,41,  33,53,  43,62,
1,23,   45,64,  15,34,  32,52,
49,68,  14,33,  18,37,  23,43,
24,44,  50,69,  1,24,   37,57,
19,38,  25,45,  44,63,  51,70,
34,54,  38,58,  13,32,  52,71,
53,72,  20,39,  6,26,   22,42,
53,73,  54,74,  55,75,  56,76,
57,77,  59,79,  60,80,  61,81,
62,82,  63,83,  64,84,  65,85,
66,86,  68,87,  69,88,  71,89,
72,90,  73,91,  74,92,  75,93,
76,94,  77,95,  78,96,  79,97,
80,98,  81,99,  82,100, 83,101,
84,102, 85,103, 86,105, 89,106,
90,107, 91,108, 92,109, 93,110,
85,104, 94,111, 95,112, 96,113,
97,114, 98,115, 99,116, 100,117,
101,118,        104,119,        105,120,        107,124,
108,125,        109,126,        110,127,        105,121,
111,128,        112,129,        105,122,        105,123,
113,130,        114,131,        115,132,        116,133,
117,134,        118,135,        120,136,        121,137,
122,138,        123,139,        125,140,        126,141,
127,142,        128,143,        130,144,        131,145,
132,146,        135,147,        136,148,        137,149,
138,150,        139,151,        140,152,        141,153,
144,154,        145,155,        147,156,        148,157,
149,158,        150,159,        151,160,        152,161,
153,162,        154,163,        155,164,        157,165,
```

```
158,166,	159,167,	160,168,	162,169,
163,170,	164,171,	165,172,	166,173,
167,174,	168,175,	169,176,	170,177,
171,178,	172,179,	173,180,	174,181,
175,182,	176,183,	178,184,	179,185,
180,186,	181,187,	182,188,	183,189,
184,190,	185,191,	186,192,	187,193,
188,194,	189,195,	190,196,	191,197,
192,198,	193,199,	194,200,	195,201,
196,202,	197,203,	198,204,	199,205,
200,206,	201,207,	203,208,	204,209,
205,210,	206,211,	207,212,	208,213,
209,214,	210,215,	211,216,	212,217,
213,218,	214,219,	215,220,	216,221,
218,222,	219,223,	220,224,	221,225,
222,226,	223,227,	224,228,	225,229,
226,230,	228,231,	231,232,	232,233,
233,234,	234,235,	0,0,	0,0,
0,0};
struct yysvf yysvec[] = {
0,	0,	0,
yycrank+1,	0,	0,
yycrank+0,	yysvec+1,	0,
yycrank+3,	0,	yyvstop+1,
yycrank+0,	0,	yyvstop+3,
yycrank+0,	0,	yyvstop+5,
yycrank+2,	0,	0,
yycrank+2,	0,	0,
yycrank+1,	0,	yyvstop+7,
yycrank+38,	0,	yyvstop+9,
yycrank+7,	yysvec+9,	yyvstop+11,
yycrank+6,	yysvec+9,	yyvstop+13,
yycrank+17,	yysvec+9,	yyvstop+15,
yycrank+1,	0,	0,
yycrank+1,	0,	0,
yycrank+1,	0,	0,
yycrank+2,	0,	0,
yycrank+4,	0,	0,
yycrank+1,	0,	0,
yycrank+1,	0,	0,
yycrank+1,	0,	0,
yycrank+13,	0,	0,
yycrank+3,	0,	0,
yycrank+3,	0,	0,
yycrank+3,	0,	0,
yycrank+2,	0,	0,
yycrank+14,	0,	0,
yycrank+5,	0,	0,
yycrank+5,	0,	0,
yycrank+19,	yysvec+9,	yyvstop+17,
yycrank+21,	yysvec+9,	yyvstop+19,
yycrank+16,	0,	0,
yycrank+1,	0,	0,
yycrank+1,	0,	0,
yycrank+1,	0,	0,
yycrank+2,	0,	0,
yycrank+6,	0,	0,
yycrank+1,	0,	0,
yycrank+1,	0,	0,
yycrank+6,	0,	yyvstop+21,
```

```
yycrank+8,    0,        0,
yycrank+16,   0,        0,
yycrank+0,    0,        yyvstop+23,
yycrank+2,    0,        0,
yycrank+4,    0,        0,
yycrank+1,    0,        0,
yycrank+34,   0,        0,
yycrank+6,    0,        0,
yycrank+41,   0,        0,
yycrank+54,   0,        0,
yycrank+58,   0,        0,
yycrank+62,   0,        0,
yycrank+17,   0,        0,
yycrank+10,   0,        0,
yycrank+9,    0,        0,
yycrank+12,   0,        0,
yycrank+13,   0,        0,
yycrank+21,   0,        0,
yycrank+7,    0,        0,
yycrank+44,   0,        0,
yycrank+16,   0,        0,
yycrank+53,   0,        0,
yycrank+22,   0,        0,
yycrank+16,   0,        0,
yycrank+33,   0,        0,
yycrank+87,   0,        0,
yycrank+94,   0,        0,
yycrank+0,    0,        yyvstop+25,
yycrank+86,   0,        0,
yycrank+89,   0,        0,
yycrank+0,    0,        yyvstop+27,
yycrank+38,   0,        0,
yycrank+30,   0,        0,
yycrank+44,   0,        0,
yycrank+37,   0,        0,
yycrank+46,   0,        0,
yycrank+47,   0,        0,
yycrank+113,  0,        0,
yycrank+66,   0,        0,
yycrank+31,   0,        0,
yycrank+51,   0,        0,
yycrank+81,   0,        0,
yycrank+40,   0,        0,
yycrank+35,   0,        0,
yycrank+120,  0,        0,
yycrank+112,  0,        0,
yycrank+122,  0,        yyvstop+29,
yycrank+0,    0,        yyvstop+31,
yycrank+0,    0,        yyvstop+33,
yycrank+41,   0,        0,
yycrank+55,   0,        0,
yycrank+58,   0,        0,
yycrank+48,   0,        0,
yycrank+50,   0,        0,
yycrank+52,   0,        0,
yycrank+78,   0,        0,
yycrank+49,   0,        0,
yycrank+59,   0,        0,
yycrank+56,   0,        0,
yycrank+100,  0,        0,
```

```
yycrank+66,   0,          0,
yycrank+67,   0,          0,
yycrank+0,    0,          yyvstop+35,
yycrank+0,    0,          yyvstop+37,
yycrank+128,  0,          0,
yycrank+102,  0,          0,
yycrank+0,    0,          yyvstop+39,
yycrank+63,   0,          0,
yycrank+56,   0,          0,
yycrank+76,   0,          0,
yycrank+73,   0,          0,
yycrank+75,   0,          0,
yycrank+57,   0,          0,
yycrank+69,   0,          0,
yycrank+73,   0,          0,
yycrank+81,   0,          0,
yycrank+94,   0,          0,
yycrank+76,   0,          0,
yycrank+145,  0,          0,
yycrank+0,    0,          yyvstop+41,
yycrank+89,   0,          0,
yycrank+77,   0,          0,
yycrank+83,   0,          0,
yycrank+88,   0,          0,
yycrank+0,    0,          yyvstop+43,
yycrank+89,   0,          0,
yycrank+75,   0,          0,
yycrank+77,   0,          0,
yycrank+78,   0,          0,
yycrank+0,    0,          yyvstop+45,
yycrank+82,   0,          0,
yycrank+90,   0,          0,
yycrank+81,   0,          0,
yycrank+0,    0,          yyvstop+47,
yycrank+0,    0,          yyvstop+49,
yycrank+82,   0,          0,
yycrank+82,   0,          0,
yycrank+81,   0,          0,
yycrank+90,   0,          0,
yycrank+86,   0,          0,
yycrank+88,   0,          0,
yycrank+98,   0,          0,
yycrank+0,    0,          yyvstop+51,
yycrank+0,    0,          yyvstop+53,
yycrank+103,  0,          0,
yycrank+83,   0,          0,
yycrank+0,    0,          yyvstop+55,
yycrank+165,  0,          0,
yycrank+110,  0,          0,
yycrank+111,  0,          0,
yycrank+102,  0,          0,
yycrank+95,   0,          0,
yycrank+96,   0,          0,
yycrank+101,  0,          0,
yycrank+99,   0,          0,
yycrank+117,  0,          0,
yycrank+0,    0,          yyvstop+57,
yycrank+183,  0,          0,
yycrank+108,  0,          0,
yycrank+185,  0,          0,
```

```
yycrank+121,  0,          0,
yycrank+0,    0,          yyvstop+59,
yycrank+109,  0,          0,
yycrank+112,  0,          0,
yycrank+105,  0,          0,
yycrank+146,  0,          0,
yycrank+118,  0,          0,
yycrank+114,  0,          0,
yycrank+122,  0,          0,
yycrank+194,  0,          0,
yycrank+106,  0,          0,
yycrank+123,  0,          0,
yycrank+124,  0,          0,
yycrank+130,  0,          0,
yycrank+120,  0,          0,
yycrank+131,  0,          0,
yycrank+167,  0,          0,
yycrank+0,    0,          yyvstop+61,
yycrank+123,  0,          0,
yycrank+125,  0,          0,
yycrank+204,  0,          0,
yycrank+121,  0,          0,
yycrank+123,  0,          0,
yycrank+122,  0,          0,
yycrank+130,  0,          0,
yycrank+134,  0,          0,
yycrank+128,  0,          0,
yycrank+211,  0,          0,
yycrank+212,  0,          0,
yycrank+130,  0,          0,
yycrank+214,  0,          0,
yycrank+215,  0,          0,
yycrank+147,  0,          0,
yycrank+152,  0,          0,
yycrank+170,  0,          0,
yycrank+130,  0,          0,
yycrank+215,  0,          0,
yycrank+169,  0,          0,
yycrank+139,  0,          0,
yycrank+223,  0,          0,
yycrank+155,  0,          0,
yycrank+225,  0,          0,
yycrank+0,    0,          yyvstop+63,
yycrank+153,  0,          0,
yycrank+147,  0,          0,
yycrank+160,  0,          0,
yycrank+151,  0,          0,
yycrank+197,  0,          0,
yycrank+154,  0,          0,
yycrank+153,  0,          0,
yycrank+168,  0,          0,
yycrank+166,  0,          0,
yycrank+151,  0,          0,
yycrank+167,  0,          0,
yycrank+159,  0,          0,
yycrank+154,  0,          0,
yycrank+166,  0,          0,
yycrank+0,    0,          yyvstop+65,
yycrank+161,  0,          0,
yycrank+158,  0,          0,
```

```
yycrank+177,  0,          0,
yycrank+165,  0,          0,
yycrank+159,  0,          0,
yycrank+176,  0,          0,
yycrank+166,  0,          0,
yycrank+176,  0,          0,
yycrank+164,  0,          0,
yycrank+0,    0,          yyvstop+67,
yycrank+173,  0,          0,
yycrank+0,    0,          yyvstop+69,
yycrank+0,    0,          yyvstop+71,
yycrank+181,  0,          0,
yycrank+163,  0,          0,
yycrank+183,  0,          0,
yycrank+171,  0,          0,
yycrank+0,    0,          yyvstop+73,
0,      0,      0};
struct yywork *yytop = yycrank+285;
struct yysvf *yybgin = yysvec+1;
char yymatch[] = {
00  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
01  ,011 ,01  ,01  ,01  ,01  ,01  ,01  ,
01  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
01  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
011 ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
01  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
'0' ,'1' ,'1' ,'1' ,'1' ,'1' ,'1' ,'1' ,
'1' ,'1' ,01  ,01  ,01  ,01  ,01  ,01  ,
01  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
01  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
01  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
01  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
01  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
01  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
01  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
0};
char yyextra[] = {
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0};
#ifndef lint
static  char ncform_sccsid[] = "@(#)ncform 1.6 88/02/08 SMI"; /* from S5R2 1.2 */
#endif

int yylineno =1;
# define YYU(x) x
# define NLSTATE yyprevious=YYNEWLINE
char yytext[YYLMAX];
struct yysvf *yylstate [YYLMAX], **yylsp, **yyolsp;
char yysbuf[YYLMAX];
char *yysptr = yysbuf;
int *yyfnd;
extern struct yysvf *yyestate;
int yyprevious = YYNEWLINE;
yylook(){
        register struct yysvf *yystate, **lsp;
        register struct yywork *yyt;
```

```
        struct yysvf *yyz;
        int yych, yyfirst;
        struct yywork *yyr;
# ifdef LEXDEBUG
        int debug;
# endif
        char *yylastch;
        /* start off machines */
# ifdef LEXDEBUG
        debug = 0;
# endif
        yyfirst=1;
        if (!yymorfg)
                yylastch = yytext;
        else {
                yymorfg=0;
                yylastch = yytext+yyleng;
                }
        for(;;){
                lsp = yylstate;
                yyestate = yystate = yybgin;
                if (yyprevious==YYNEWLINE) yystate++;
                for (;;){
# ifdef LEXDEBUG
                        if(debug)fprintf(yyout,"state %d\n",yystate-yysvec-1);
# endif
                        yyt = yystate->yystoff;
                        if(yyt == yycrank && !yyfirst){  /* may not be any transitions */
                                yyz = yystate->yyother;
                                if(yyz == 0)break;
                                if(yyz->yystoff == yycrank)break;
                                }
                        *yylastch++ = yych = input();
                        yyfirst=0;
                tryagain:
# ifdef LEXDEBUG
                        if(debug){
                                fprintf(yyout,"char ");
                                allprint(yych);
                                putchar('\n');
                                }
# endif
                        yyr = yyt;
                        if ( (int)yyt > (int)yycrank){
                                yyt = yyr + yych;
                                if (yyt <= yytop && yyt->verify+yysvec == yystate){
                                        if(yyt->advance+yysvec == YYLERR)       /* error
transitions */
                                                {unput(*--yylastch);break;}
                                        *lsp++ = yystate = yyt->advance+yysvec;
                                        goto contin;
                                        }
                                }
# ifdef YYOPTIM
                        else if((int)yyt < (int)yycrank) {              /* r < yycrank */
                                yyt = yyr = yycrank+(yycrank-yyt);
# ifdef LEXDEBUG
                                if(debug)fprintf(yyout,"compressed state\n");
# endif
                                yyt = yyt + yych;
```

```
                        if(yyt <= yytop && yyt->verify+yysvec == yystate){
                                if(yyt->advance+yysvec == YYLERR)       /* error
transitions */
                                        {unput(*--yylastch);break;}
                                *lsp++ = yystate = yyt->advance+yysvec;
                                goto contin;
                                }
                        yyt = yyr + YYU(yymatch[yych]);
# ifdef LEXDEBUG
                        if(debug){
                                fprintf(yyout,"try fall back character ");
                                allprint(YYU(yymatch[yych]));
                                putchar('\n');
                                }
# endif
                        if(yyt <= yytop && yyt->verify+yysvec == yystate){
                                if(yyt->advance+yysvec == YYLERR)       /* error
transition */
                                        {unput(*--yylastch);break;}
                                *lsp++ = yystate = yyt->advance+yysvec;
                                goto contin;
                                }
                        }
                if ((yystate = yystate->yyother) && (yyt= yystate->yystoff) != yycrank){
# ifdef LEXDEBUG
                        if(debug)fprintf(yyout,"fall back to state %d\n",yystate-yysvec-1);
# endif
                        goto tryagain;
                        }
# endif
                else
                        {unput(*--yylastch);break;}
        contin:
# ifdef LEXDEBUG
                if(debug){
                        fprintf(yyout,"state %d char ",yystate-yysvec-1);
                        allprint(yych);
                        putchar('\n');
                        }
# endif
                ;
                }
# ifdef LEXDEBUG
        if(debug){
                fprintf(yyout,"stopped at %d with ",*(lsp-1)-yysvec-1);
                allprint(yych);
                putchar('\n');
                }
# endif
        while (lsp-- > yylstate){
                *yylastch-- = 0;
                if (*lsp != 0 && (yyfnd= (*lsp)->yystops) && *yyfnd > 0){
                        yyolsp = lsp;
                        if(yyextra[*yyfnd]){            /* must backup */
                                while(yyback((*lsp)->yystops,-*yyfnd) != 1 && lsp >
yylstate){
                                        lsp--;
                                        unput(*yylastch--);
                                        }
                                }
```

```
                        yyprevious = YYU(*yylastch);
                        yylsp = lsp;
                        yyleng = yylastch-yytext+1;
                        yytext[yyleng] = 0;
# ifdef LEXDEBUG
                        if(debug){
                                fprintf(yyout,"\nmatch ");
                                sprint(yytext);
                                fprintf(yyout," action %d\n",*yyfnd);
                                }
# endif
                        return(*yyfnd++);
                        }
                unput(*yylastch);
                }
        if (yytext[0] == 0  /* && feof(yyin) */)
                {
                yysptr=yysbuf;
                return(0);
                }
        yyprevious = yytext[0] = input();
        if (yyprevious>0)
                output(yyprevious);
        yylastch=yytext;
# ifdef LEXDEBUG
        if(debug)putchar('\n');
# endif
        }
    }
yyback(p, m)
    int *p;
{
if (p==0) return(0);
while (*p)
    {
    if (*p++ == m)
        return(1);
    }
return(0);
}
    /* the following are only used in the lex library */
yyinput(){
    return(input());
    }
yyoutput(c)
  int c; {
    output(c);
    }
yyunput(c)
   int c; {
    unput(c);
    }
```

```
#ifndef lint
static char yysccsid[] = "@(#)yaccpar     1.7 (Berkeley) 09/09/90";
#endif
#define YYBYACC 1
#line 2 "com2yacc.yy"

# include <h:\users\lbw\c600\include\stdio.h>
# include <h:\users\lbw\c600\include\string.h>

# define TRUE 1
# define FALSE 0
# define DBUFFER_LENGTH 28
# define DTRANS_LENGTH 179
# define DRECEIVE_LENGTH 179
# define S_IWRITE 0000200 /* write permission, owner */

extern char xplexer_dataplexer[7];
extern char *datetime;
char path_name[80];
int loop;
int handle;
long int length;
int count;
char *buffer_trans;
char *buffer_receive;
char buffer[DBUFFER_LENGTH];
char buffer_temp_trans[59];
char buffer_temp_receive[59];

FILE *fp;

struct input_stack { char inputChar[256];
                     int top;
                   } input_stackptr;

#line 37 "y_tab.c"
#define NUMB 257
#define NEW_LINE 258
#define COMMENT 259
#define TRANS_COMMENT 260
#define REC_COMMENT 261
#define BUFFER_COMMENT 262
#define UTIL_COMMENT 263
#define CHAN8 264
#define CHAN16 265
#define CHAN24 266
#define CHAN32 267
#define NODE 268
#define MIN_COMMENT 269
#define CHANNEL_COMMENT 270
#define TIMES_ONE_HUNDRED 271
#define CHARACTERS_COMMENT 272
#define TIMES_ONE_THOUSAND 273
#define DEST_BUSY 274
#define DATA_LINK_TIMEOUT 275
#define INVALID_RESPONSE 276
#define LINK_NOT_DATAPLEXER 277
#define END_CHANNEL_COMMENT 278
#define NOT_PROPERLY_CONFIGURED 279
#define MESSAGES_PENDING 280
```

```
#define STANDBY 281
#define YYERRCODE 256
short yylhs[] = {                                        -1,
    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
    0,    0,    1,    0,    2,    0,    3,    0,    4,    0,
    5,    0,
};
short yylen[] = {                                         2,
    0,    2,    3,    2,    2,    1,    1,    1,    1,    1,
    1,    2,    4,    1,    6,    3,    5,    4,    4,    5,
    3,    3,    0,   11,    0,   12,    0,   12,    0,   12,
    0,   12,
};
short yydefred[] = {                                      0,
   14,    0,    0,    0,    0,    0,    0,    0,    0,    0,
    0,    0,    0,    0,    7,    8,    9,   10,    0,   11,
    0,    6,    0,    0,    2,    4,    0,    0,    0,    0,
    0,    0,    0,    0,    0,    0,    0,    0,    0,   12,
    5,    3,    0,    0,    0,    0,   16,    0,    0,    0,
    0,    0,    0,   21,   22,    0,   18,   19,    0,    0,
   13,    0,    0,    0,    0,    0,    0,   17,    0,   20,
    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
   15,    0,    0,    0,    0,    0,    0,    0,    0,    0,
    0,   23,    0,    0,    0,    0,    0,   25,   27,   29,
   31,   24,    0,    0,    0,    0,   26,   28,   30,   32,
};
short yydgoto[] = {                                      23,
   97,  103,  104,  105,  106,
};
short yysindex[] = {                                   -151,
    0, -188, -151, -255, -254, -251, -249, -252, -250, -248,
 -247, -245, -261, -258,    0,    0,    0,    0, -240,    0,
 -151,    0,    0, -151,    0,    0, -253, -239, -237, -151,
 -236, -234, -233, -232, -231, -230, -151, -151, -228,    0,
    0,    0, -151, -151, -227, -226,    0, -225, -224, -223,
 -220, -219, -218,    0,    0, -217,    0,    0, -151, -214,
    0, -151, -213, -212, -211, -210, -209,    0, -207,    0,
 -206, -205, -204, -203, -151, -202, -201, -199, -198, -196,
    0, -195, -194, -193, -192, -191, -190, -174, -172, -163,
 -161,    0, -160, -159, -158, -157, -151,    0,    0,    0,
    0,    0, -151, -151, -151, -151,    0,    0,    0,    0,
};
short yyrindex[] = {                                     48,
    0,   48,   48,    0,    0,    0,    0,    0,    0,    0,
    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
   48,    0,    0,   48,    0,    0,    0,    0,    0,   48,
    0,    0,    0,    0,    0,    0,   48,   48,    0,    0,
    0,    0,   48,   48,    0,    0,    0,    0,    0,    0,
    0,    0,    0,    0,    0,    0,    0,    0,   48,    0,
    0,   48,    0,    0,    0,    0,    0,    0,    0,    0,
    0,    0,    0,    0,   48,    0,    0,    0,    0,    0,
    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
    0,    0,    0,    0,    0,    0,   48,    0,    0,    0,
    0,    0,   48,   48,   48,   48,    0,    0,    0,    0,
};
short yygindex[] = {                                     -2,
    0,    0,    0,    0,    0,
};
#define YYTABLESIZE 130
```

```
short yytable[] = {                                         25,
   26,   27,   29,   43,   44,   31,   30,   32,   33,   38,
   34,   37,   35,   36,   39,   28,   40,   45,   41,   46,
   48,   42,   49,   50,   51,   52,   53,   47,   56,   59,
   60,   61,   62,   63,   54,   55,   64,   65,   66,   67,
   57,   58,   69,   71,   72,   73,   74,    1,   75,   76,
   77,   78,   79,   80,   82,   83,   68,   84,   85,   70,
   86,   87,   88,   89,   90,   91,   92,    1,    2,    3,
    4,    5,   81,    6,    7,    8,    9,   10,   11,   12,
   24,   13,   93,   14,   94,   15,   16,   17,   18,   19,
   20,   21,   22,   95,  102,   96,   98,   99,  100,  101,
  107,  108,  109,  110,    1,    2,    3,    4,    5,    0,
    6,    7,    8,    9,   10,   11,   12,    0,   13,    0,
   14,    0,   15,   16,   17,   18,   19,   20,   21,   22,
};
short yycheck[] = {                                          2,
    3,  257,  257,  257,  258,  257,  261,  257,  261,  271,
  261,  257,  261,  261,  273,  271,  257,  257,   21,  257,
  257,   24,  257,  257,  257,  257,  257,   30,  257,  257,
  257,  257,  257,  257,   37,   38,  257,  257,  257,  257,
   43,   44,  257,  257,  257,  257,  257,    0,  258,  257,
  257,  257,  257,  257,  257,  257,   59,  257,  257,   62,
  257,  257,  257,  257,  257,  257,  257,  256,  257,  258,
  259,  260,   75,  262,  263,  264,  265,  266,  267,  268,
  269,  270,  257,  272,  257,  274,  275,  276,  277,  278,
  279,  280,  281,  257,   97,  257,  257,  257,  257,  257,
  103,  104,  105,  106,  256,  257,  258,  259,  260,   -1,
  262,  263,  264,  265,  266,  267,  268,   -1,  270,   -1,
  272,   -1,  274,  275,  276,  277,  278,  279,  280,  281,
};
#define YYFINAL 23
#ifndef YYDEBUG
#define YYDEBUG 0
#endif
#define YYMAXTOKEN 281
#if YYDEBUG
char *yyname[] = {
"end-of-file",0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,"NUMB";"NEW_LINE","COMMENT",
"TRANS_COMMENT","REC_COMMENT";"BUFFER_COMMENT","UTIL_COMMENT","CHAN
8","CHAN16",
"CHAN24","CHAN32","NODE","MIN_COMMENT","CHANNEL_COMMENT","TIMES_ONE_HU
NDRED",
"CHARACTERS_COMMENT","TIMES_ONE_THOUSAND","DEST_BUSY","DATA_LINK_TIM
EOUT",
"INVALID_RESPONSE","LINK_NOT_DATAPLEXER","END_CHANNEL_COMMENT",
"NOT_PROPERLY_CONFIGURED","MESSAGES_PENDING","STANDBY",
};
char *yyrule[] = {
"$accept : dataplexer",
"dataplexer :",
"dataplexer : NUMB dataplexer",
"dataplexer : NUMB MIN_COMMENT dataplexer",
"dataplexer : NEW_LINE dataplexer",
"dataplexer : MESSAGES_PENDING dataplexer",
```

```
"dataplexer : STANDBY",
"dataplexer : DEST_BUSY",
"dataplexer : DATA_LINK_TIMEOUT",
"dataplexer : INVALID_RESPONSE",
"dataplexer : LINK_NOT_DATAPLEXER",
"dataplexer : NOT_PROPERLY_CONFIGURED",
"dataplexer : END_CHANNEL_COMMENT NUMB",
"dataplexer : BUFFER_COMMENT NUMB NUMB NUMB",
"dataplexer : error",
"dataplexer : CHARACTERS_COMMENT TIMES_ONE_THOUSAND NUMB NUMB
NEW_LINE dataplexer",
"dataplexer : TRANS_COMMENT REC_COMMENT dataplexer",
"dataplexer : COMMENT TIMES_ONE_HUNDRED NUMB NUMB dataplexer",
"dataplexer : COMMENT NUMB NUMB dataplexer",
"dataplexer : COMMENT NUMB NEW_LINE dataplexer",
"dataplexer : UTIL_COMMENT NUMB NUMB NUMB dataplexer",
"dataplexer : NODE NUMB dataplexer",
"dataplexer : CHANNEL_COMMENT TIMES_ONE_HUNDRED dataplexer",
"$$1 :",
"dataplexer : TRANS_COMMENT NUMB NUMB NUMB NUMB NUMB NUMB NUMB NUMB
$$1 dataplexer",
"$$2 :",
"dataplexer : CHAN8 REC_COMMENT NUMB NUMB NUMB NUMB NUMB NUMB NUMB
NUMB $$2 dataplexer",
"$$3 :",
"dataplexer : CHAN16 REC_COMMENT NUMB NUMB NUMB NUMB NUMB NUMB NUMB
NUMB $$3 dataplexer",
"$$4 :",
"dataplexer : CHAN24 REC_COMMENT NUMB NUMB NUMB NUMB NUMB NUMB NUMB
NUMB $$4 dataplexer",
"$$5 :",
"dataplexer : CHAN32 REC_COMMENT NUMB NUMB NUMB NUMB NUMB NUMB NUMB
NUMB $$5 dataplexer",
};
#endif
#ifndef YYSTYPE
typedef int YYSTYPE;
#endif
#define yyclearin (yychar=(-1))
#define yyerrok (yyerrflag=0)
#ifdef YYSTACKSIZE
#ifndef YYMAXDEPTH
#define YYMAXDEPTH YYSTACKSIZE
#endif
#else
#ifdef YYMAXDEPTH
#define YYSTACKSIZE YYMAXDEPTH
#else
#define YYSTACKSIZE 600
#define YYMAXDEPTH 600
#endif
#endif
int yydebug;
int yynerrs;
int yyerrflag;
int yychar;
short *yyssp;
YYSTYPE *yyvsp;
YYSTYPE yyval;
YYSTYPE yylval;
```

```
short yyss[YYSTACKSIZE];
YYSTYPE yyvs[YYSTACKSIZE];
#define yystacksize YYSTACKSIZE
#line 196 "com2yacc.yy"

# include "LEXYY.C"

yyerror(s)
  char *s;
{
  fprintf(stderr, "Error: %s\n", s);
}


yywrap()
{
  return 1;
}

char *
append_file_name_to_path(file_name)
  char file_name[12];
{
  char new_path[268];
  strcpy(new_path, path_name);
  strcat(new_path, file_name);
  return(new_path);
}
#line 261 "y_tab.c"
#define YYABORT goto yyabort
#define YYACCEPT goto yyaccept
#define YYERROR goto yyerrlab
int
yyparse()
{
    register int yym, yyn, yystate;
#if YYDEBUG
    register char *yys;
    extern char *getenv();

    if (yys = getenv("YYDEBUG"))
    {
        yyn = *yys;
        if (yyn >= '0' && yyn <= '9')
            yydebug = yyn - '0';
    }
#endif

    yynerrs = 0;
    yyerrflag = 0;
    yychar = (-1);

    yyssp = yyss;
    yyvsp = yyvs;
    *yyssp = yystate = 0;

yyloop:
    if (yyn = yydefred[yystate]) goto yyreduce;
    if (yychar < 0)
    {
```

```
        if ((yychar = yylex()) < 0) yychar = 0;
#if YYDEBUG
        if (yydebug)
        {
            yys = 0;
            if (yychar <= YYMAXTOKEN) yys = yyname[yychar];
            if (!yys) yys = "illegal-symbol";
            printf("yydebug: state %d, reading %d (%s)\n", yystate,
                    yychar, yys);
        }
#endif
    }
    if ((yyn = yysindex[yystate]) && (yyn += yychar) >= 0 &&
            yyn <= YYTABLESIZE && yycheck[yyn] == yychar)
    {
#if YYDEBUG
        if (yydebug)
            printf("yydebug: state %d, shifting to state %d\n",
                    yystate, yytable[yyn]);
#endif
        if (yyssp >= yyss + yystacksize - 1)
        {
            goto yyoverflow;
        }
        *++yyssp = yystate = yytable[yyn];
        *++yyvsp = yylval;
        yychar = (-1);
        if (yyerrflag > 0)  --yyerrflag;
        goto yyloop;
    }
    if ((yyn = yyrindex[yystate]) && (yyn += yychar) >= 0 &&
            yyn <= YYTABLESIZE && yycheck[yyn] == yychar)
    {
        yyn = yytable[yyn];
        goto yyreduce;
    }
    if (yyerrflag) goto yyinrecovery;
#ifdef lint
    goto yynewerror;
#endif
yynewerror:
    yyerror("syntax error");
#ifdef lint
    goto yyerrlab;
#endif
yyerrlab:
    ++yynerrs;
yyinrecovery:
    if (yyerrflag < 3)
    {
        yyerrflag = 3;
        for (;;)
        {
            if ((yyn = yysindex[*yyssp]) && (yyn += YYERRCODE) >= 0 &&
                    yyn <= YYTABLESIZE && yycheck[yyn] == YYERRCODE)
            {
#if YYDEBUG
                if (yydebug)
                    printf("yydebug: state %d, error recovery shifting\
 to state %d\n", *yyssp, yytable[yyn]);
```

```
#endif
                if (yyssp >= yyss + yystacksize - 1)
                {
                    goto yyoverflow;
                }
                *++yyssp = yystate = yytable[yyn];
                *++yyvsp = yylval;
                goto yyloop;
            }
            else
            {
#if YYDEBUG
                if (yydebug)
                    printf("yydebug: error recovery discarding state %d\n",
                            *yyssp);
#endif
                if (yyssp <= yyss) goto yyabort;
                --yyssp;
                --yyvsp;
            }
        }
    }
    else
    {
        if (yychar == 0) goto yyabort;
#if YYDEBUG
        if (yydebug)
        {
            yys = 0;
            if (yychar <= YYMAXTOKEN) yys = yyname[yychar];
            if (!yys) yys = "illegal-symbol";
            printf("yydebug: state %d, error recovery discards token %d (%s)\n",
                    yystate, yychar, yys);
        }
#endif
        yychar = (-1);
        goto yyloop;
    }
yyreduce:
#if YYDEBUG
    if (yydebug)
        printf("yydebug: state %d, reducing by rule %d (%s)\n",
                yystate, yyn, yyrule[yyn]);
#endif
    yym = yylen[yyn];
    yyval = yyvsp[1-yym];
    switch (yyn)
    {
case 6:
#line 55 "com2yacc.yy"
{ yyerrok; YYACCEPT; }
break;
case 7:
#line 57 "com2yacc.yy"
{ yyerrok; YYACCEPT; }
break;
case 8:
#line 59 "com2yacc.yy"
{ yyerrok; YYACCEPT; }
break;
```

```
case 9:
#line 61 "com2yacc.yy"
{ yyerrok; YYACCEPT; }
break;
case 10:
#line 63 "com2yacc.yy"
{ yyerrok; YYACCEPT; }
break;
case 11:
#line 65 "com2yacc.yy"
{ yyerrok; YYACCEPT; }
break;
case 12:
#line 67 "com2yacc.yy"
{ yyerrok; YYACCEPT; }
break;
case 13:
#line 71 "com2yacc.yy"
{

                    /* return from the parser in order to continue
                       interrogating the network for more data or
                       to wait for the next interrogation period. */

              sprintf(buffer, "%s%s%3d%3d%3d\n", datetime, xplexer_dataplexer, yyvsp[-2],
yyvsp[-1], yyvsp[0]);
              while((handle = sopen(append_file_name_to_path("DBUFFER.TXT"), O_CREAT |
O_WRONLY, SH_DENYNO, S_IWRITE)) == FALSE);
              length = filelength(handle) + (long)DBUFFER_LENGTH + 1;
              while(locking(handle, LK_NBRLCK, length) != -1);
              lseek(handle, 0L, SEEK_END);
              write(handle, buffer, DBUFFER_LENGTH);
              lseek(handle, 0L, SEEK_SET);
              while(locking(handle, LK_UNLCK, length) != -1);
              close(handle);

              sprintf(buffer_temp_receive, "%5u%5u%5u%5u%5u%5u%5u%5u", 0, 0, 0, 0, 0, 0, 0,
0);
              sprintf(buffer_temp_trans, "%5u%5u%5u%5u%5u%5u%5u%5u", 0, 0, 0, 0, 0, 0, 0,
0);
                  switch (count) {
                    case 2: for(loop = 1; loop <= 2; loop++)
                              { strcat(buffer_receive, buffer_temp_receive);
                                strcat(buffer_trans, buffer_temp_trans); }
                              sprintf(buffer_temp_receive, "%5u%5u%5u%5u%5u%5u%5u%5u\n",
0, 0, 0, 0, 0, 0, 0, 0);
                              sprintf(buffer_temp_trans, "%5u%5u%5u%5u%5u%5u%5u%5u\n", 0,
0, 0, 0, 0, 0, 0, 0);
                            break;
                    case 3:  strcat(buffer_receive, buffer_temp_receive);
                             strcat(buffer_trans, buffer_temp_trans);
                             sprintf(buffer_temp_receive, "%5u%5u%5u%5u%5u%5u%5u%5u\n",
0, 0, 0, 0, 0, 0, 0, 0);
                             sprintf(buffer_temp_trans, "%5u%5u%5u%5u%5u%5u%5u%5u\n", 0,
0, 0, 0, 0, 0, 0, 0);
                             break;
                    case 4: sprintf(buffer_temp_receive, "%5u%5u%5u%5u%5u%5u%5u%5u\n", 0,
0, 0, 0, 0, 0, 0, 0);
                            sprintf(buffer_temp_trans, "%5u%5u%5u%5u%5u%5u%5u%5u\n", 0, 0,
0, 0, 0, 0, 0, 0);
```

```
                        break;
                    case 5: break;
                    default: count = 1;
                             break;
                    }
                  strcat(buffer_receive, buffer_temp_receive);
                  strcat(buffer_trans, buffer_temp_trans);
                  while((handle = sopen(append_file_name_to_path("DRECEIVE.TXT"),
O_CREAT | O_WRONLY, SH_DENYNO, S_IWRITE)) == FALSE);
                  length = filelength(handle) + (long)DRECEIVE_LENGTH + 1;
                  while(locking(handle, LK_NBRLCK, length) != -1);
                  lseek(handle, 0L, SEEK_END);
                  write(handle, buffer_receive, DRECEIVE_LENGTH);
                  lseek(handle, 0L, SEEK_SET);
                  while(locking(handle, LK_UNLCK, length) != -1);
                  close(handle);

              while((handle = sopen(append_file_name_to_path("DTRANS.TXT"), O_CREAT |
O_WRONLY, SH_DENYNO, S_IWRITE)) == FALSE);
                  length = filelength(handle) + (long)DTRANS_LENGTH + 1;
                  while(locking(handle, LK_NBRLCK, length) != -1);
                  lseek(handle, 0L, SEEK_END);
                  write(handle, buffer_trans, DTRANS_LENGTH);
                  lseek(handle, 0L, SEEK_SET);
                  while(locking(handle, LK_UNLCK, length) != -1);
                  close(handle);
              count = 1;
              YYACCEPT;
              }
break;
case 14:
#line 131 "com2yacc.yy"
{ yyerrok; YYACCEPT; }
break;
case 23:
#line 151 "com2yacc.yy"
{
              switch (count) {
                case 1: sprintf(buffer_trans, "%s%s%5u%5u%5u%5u%5u%5u%5u%5u", datetime,
xplexer_dataplexer, yyvsp[-7], yyvsp[-5], yyvsp[-4], yyvsp[-3], yyvsp[-2], yyvsp[-1], yyvsp[0]);
                        count++;
                        break;
                case 4: sprintf(buffer_temp_trans, "%5u%5u%5u%5u%5u%5u%5u%5u\n", yyvsp[-
7], yyvsp[-6], yyvsp[-5], yyvsp[-4], yyvsp[-3], yyvsp[-2], yyvsp[-1], yyvsp[0]);
                        strcat(buffer_trans, buffer_temp_trans);
                        count++;
                        break;
                default: sprintf(buffer_temp_trans, "%5u%5u%5u%5u%5u%5u%5u%5u", yyvsp[-7],
yyvsp[-6], yyvsp[-5], yyvsp[-4], yyvsp[-3], yyvsp[-2], yyvsp[-1], yyvsp[0]);
                         strcat(buffer_trans, buffer_temp_trans);
                         count++;
                         break;
                }
              }
break;
case 25:
#line 169 "com2yacc.yy"
{
```

```
                    sprintf(buffer_receive, "%s%s%5u%5u%5u%5u%5u%5u%5u%5u", datetime,
xplexer_dataplexer, yyvsp[-7], yyvsp[-6], yyvsp[-5], yyvsp[-4], yyvsp[-3], yyvsp[-2], yyvsp[-1],
yyvsp[0]);
                }
break;
case 27:
#line 175 "com2yacc.yy"
{
                    sprintf(buffer_temp_receive, "%5u%5u%5u%5u%5u%5u%5u%5u", yyvsp[-7],
yyvsp[-6], yyvsp[-5], yyvsp[-4], yyvsp[-3], yyvsp[-2], yyvsp[-1], yyvsp[0]);
                    strcat(buffer_receive, buffer_temp_receive);
                }
break;
case 29:
#line 182 "com2yacc.yy"
{
                    sprintf(buffer_temp_receive, "%5u%5u%5u%5u%5u%5u%5u%5u", yyvsp[-7],
yyvsp[-6], yyvsp[-5], yyvsp[-4], yyvsp[-3], yyvsp[-2], yyvsp[-1], yyvsp[0]);
                    strcat(buffer_receive, buffer_temp_receive);
                }
break;
case 31:
#line 189 "com2yacc.yy"
{
                    sprintf(buffer_temp_receive, "%5u%5u%5u%5u%5u%5u%5u%5u\n", yyvsp[-7],
yyvsp[-6], yyvsp[-5], yyvsp[-4], yyvsp[-3], yyvsp[-2], yyvsp[-1], yyvsp[0]);
                    strcat(buffer_receive, buffer_temp_receive);
                }
break;
#line 540 "y_tab.c"
    }
    yyssp -= yym;
    yystate = *yyssp;
    yyvsp -= yym;
    yym = yylhs[yyn];
    if (yystate == 0 && yym == 0)
    {
#if YYDEBUG
        if (yydebug)
            printf("yydebug: after reduction, shifting from state 0 to\
 state %d\n", YYFINAL);
#endif
        yystate = YYFINAL;
        *++yyssp = YYFINAL;
        *++yyvsp = yyval;
        if (yychar < 0)
        {
            if ((yychar = yylex()) < 0) yychar = 0;
#if YYDEBUG
            if (yydebug)
            {
                yys = 0;
                if (yychar <= YYMAXTOKEN) yys = yyname[yychar];
                if (!yys) yys = "illegal-symbol";
                printf("yydebug: state %d, reading %d (%s)\n",
                        YYFINAL, yychar, yys);
            }
#endif
        }
        if (yychar == 0) goto yyaccept;
```

```
        goto yyloop;
    }
    if ((yyn = yygindex[yym]) && (yyn += yystate) >= 0 &&
            yyn <= YYTABLESIZE && yycheck[yyn] == yystate)
        yystate = yytable[yyn];
    else
        yystate = yydgoto[yym];
#if YYDEBUG
    if (yydebug)
        printf("yydebug: after reduction, shifting from state %d \
to state %d\n", *yyssp, yystate);
#endif
    if (yyssp >= yyss + yystacksize - 1)
    {
        goto yyoverflow;
    }
    *++yyssp = yystate;
    *++yyvsp = yyval;
    goto yyloop;
yyoverflow:
    yyerror("yacc stack overflow");
yyabort:
    return (1);
yyaccept:
    return (0);
}
```

5.1 Database File Structures

The structure of the database files can be shown in their three separate sections. These are :

3.1.1 Dataplexer Files.
3.1.2 Xplexer Files.
3.1.3 Network Configuration Files.

Note:
Not all of the following database files are used. They are included for future needs of the Network Monitor if the data is required to be analysed.

5.1.1 Dataplexer Files.

- **Buffer Utilization**

DBUFFER.DBF

Num	Field Name	Field Type	Width	Dec	Index
1	DATE	Date	8		Y
2	HOUR	Character	2		Y
3	MINUTES	Character	2		Y
4	XPLEXER	Character	3		Y
5	DATAPLEXER	Character	3		Y
6	AVERAGE	Numeric	3	0	N
7	PEAK	Numeric	3	0	N
8	LAST	Numeric	3	0	N

- **Channel Throughput**

DTRANS.DBF

Num	Field Name	Field Type	Width	Dec	Index
1	DATE	Date	8		Y
2	HOUR	Character	2		Y
3	MINUTES	Character	2		Y
4	XPLEXER	Character	3		Y
5	DATAPLEXER	Character	3		Y
6	CHAN0	Numeric	5	0	N
7	CHAN1	Numeric	5	0	N
8	CHAN2	Numeric	5	0	N
9	CHAN3	Numeric	5	0	N
10	CHAN4	Numeric	5	0	N
11	CHAN5	Numeric	5	0	N
12	CHAN6	Numeric	5	0	N
13	CHAN7	Numeric	5	0	N
14	CHAN8	Numeric	5	0	N
15	CHAN9	Numeric	5	0	N
16	CHAN10	Numeric	5	0	N
17	CHAN11	Numeric	5	0	N
18	CHAN12	Numeric	5	0	N
19	CHAN13	Numeric	5	0	N
20	CHAN14	Numeric	5	0	N
21	CHAN15	Numeric	5	0	N
22	CHAN16	Numeric	5	0	N
23	CHAN17	Numeric	5	0	N
24	CHAN18	Numeric	5	0	N
25	CHAN19	Numeric	5	0	N
26	CHAN20	Numeric	5	0	N
27	CHAN21	Numeric	5	0	N
28	CHAN22	Numeric	5	0	N
29	CHAN23	Numeric	5	0	N
30	CHAN24	Numeric	5	0	N
31	CHAN25	Numeric	5	0	N
32	CHAN26	Numeric	5	0	N
33	CHAN27	Numeric	5	0	N
34	CHAN28	Numeric	5	0	N
35	CHAN29	Numeric	5	0	N
36	CHAN30	Numeric	5	0	N
37	CHAN31	Numeric	5	0	N

DRECEIVE.DBF

Num	Field Name	Field Type	Width	Dec	Index
1	DATE	Date	8		Y
2	HOUR	Character	2		Y
3	MINUTES	Character	2		Y
4	XPLEXER	Character	3		Y
5	DATAPLEXER	Character	3		Y
6	CHAN0	Numeric	5	0	N
7	CHAN1	Numeric	5	0	N
8	CHAN2	Numeric	5	0	N
9	CHAN3	Numeric	5	0	N
10	CHAN4	Numeric	5	0	N
11	CHAN5	Numeric	5	0	N
12	CHAN6	Numeric	5	0	N
13	CHAN7	Numeric	5	0	N
14	CHAN8	Numeric	5	0	N
15	CHAN9	Numeric	5	0	N
16	CHAN10	Numeric	5	0	N
17	CHAN11	Numeric	5	0	N
18	CHAN12	Numeric	5	0	N
19	CHAN13	Numeric	5	0	N
20	CHAN14	Numeric	5	0	N
21	CHAN15	Numeric	5	0	N
22	CHAN16	Numeric	5	0	N
23	CHAN17	Numeric	5	0	N
24	CHAN18	Numeric	5	0	N
25	CHAN19	Numeric	5	0	N
26	CHAN20	Numeric	5	0	N
27	CHAN21	Numeric	5	0	N
28	CHAN22	Numeric	5	0	N
29	CHAN23	Numeric	5	0	N
30	CHAN24	Numeric	5	0	N
31	CHAN25	Numeric	5	0	N
32	CHAN26	Numeric	5	0	N
33	CHAN27	Numeric	5	0	N
34	CHAN28	Numeric	5	0	N
35	CHAN29	Numeric	5	0	N
36	CHAN30	Numeric	5	0	N
37	CHAN31	Numeric	5	0	N

5.1.2 XPLEXER FILES.

CONVERT.DBF

Num	Field Name	Field Type	Width	Dec	Index
1	EVENTNUMB	Character	2		Y
2	EVENTCODE	Character	4		Y

XPLEX01.DBF

Num	Field Name	Field Type	Width	Dec	Index
1	EVENTCODE	Character	4		Y
2	EVENTDESCR	Character	41		N

XPLEX02.TXT

Num	Field Name	Field Type	Width	Dec	Index
1	DATE	Date	8		Y
2	HOUR	Character	2		Y
3	MINUTES	Character	2		Y
4	COUNT	Character	3		N
5	DEVICETYPE	Character	1		N
6	EVENTNUMB	Character	2		N
7	NODE	Character	3		Y

XPLEX03.TXT

Num	Field Name	Field Type	Width	Dec	Index
1	DATE	Date	8		Y
2	HOUR	Character	2		Y
3	MINUTES	Character	2		Y
4	COUNT	Character	3		N
5	DEVICETYPE	Character	1		N
6	EVENTNUMB	Character	2		N
7	NODE	Character	3		Y
8	LINK	Character	3		Y

XPLEX04.TXT

Num	Field Name	Field Type	Width	Dec	Index
1	DATE	Date	8		Y
2	HOUR	Character	2		Y
3	MINUTES	Character	2		Y
4	COUNT	Character	3		N
5	DEVICETYPE	Character	1		N
6	EVENTNUMB	Character	2		N
7	NODE	Character	3		Y
8	DIALSTRING	Character	20		N

XPLEX05.TXT

Num	Field Name	Field Type	Width	Dec	Index
1	DATE	Date	8		Y
2	HOUR	Character	2		Y
3	MINUTES	Character	2		Y
4	COUNT	Character	3		N
5	DEVICETYPE	Character	1		N
6	EVENTNUMB	Character	2		Y
7	NODE	Character	3		Y
8	LINK	Character	3		Y
9	CHANNEL	Character	3		Y

XPLEX06.TXT

Num	Field Name	Field Type	Width	Dec	Index
1	DATE	Date	8		Y
2	HOUR	Character	2		Y
3	MINUTES	Character	2		Y
4	COUNT	Character	3		N
5	DEVICETYPE	Character	1		N
6	EVENTNUMB	Character	2		N
7	NODE	Character	3		Y
8	LINK	Character	3		Y
9	CHANNEL	Character	3		Y
10	DIALSTRING	Character	20		N

- **Queue Statistics**

XPLEX07.TXT

Num	Field Name	Field Type	Width	Dec	Index
1	DATE	Date	8		Y
2	HOUR	Character	2		Y
3	MINUTES	Character	2		Y
4	COUNT	Character	3		N
5	DEVICETYPE	Character	1		N
6	EVENTNUMB	Character	2		N
7	NODE	Character	3		Y
8	LINK	Character	3		Y
9	CHANNEL	Character	3		Y
10	QUEUEPLAC E	Numeric	2	0	N

XPLEX08.TXT

Num	Field Name	Field Type	Width	Dec	Index
1	DATE	Date	8		Y
2	HOUR	Character	2		Y
3	MINUTES	Character	2		Y
4	COUNT	Character	3		N
5	DEVICETYPE	Character	1		N
6	EVENTNUMB	Character	2		N
7	NODE	Character	3		Y
8	LINK	Character	3		Y
9	CHANNEL	Character	3		Y
10	TRANSMIT	Numeric	5	0	N
11	RECEIVE	Numeric	5	0	N

- **Node Statistics**

XPLEX10.TXT

Num	Field Name	Field Type	Width	Dec	Index
1	DATE	Date	8		Y
2	HOUR	Character	2		Y
3	MINUTES	Character	2		Y
4	COUNT	Character	3		N
5	DEVICETYPE	Character	1		N
6	NODE	Character	3		Y
7	MINS	Numeric	5	0	N
8	EVENTNO19	Character	2		N
9	AVERAGE19	Numeric	3	0	N
10	PEAK19	Numeric	3	0	N
11	LAST19	Numeric	3	0	N
12	EVENTNO20	Character	2		N
13	AVERAGE20	Numeric	3	0	N
14	PEAK20	Numeric	3	0	N
15	LAST20	Numeric	3	0	N
16	EVENTNO21	Character	2		N
17	AVERAGE21	Numeric	3	0	N
18	PEAK21	Numeric	3	0	N
19	LAST21	Numeric	3	0	N
20	EVENTNOLC0	Character	2		N
21	LC0	Character	1		N
22	AVERAGELC0	Numeric	3	0	N
23	PEAKLC0	Numeric	3	0	N
24	LASTLC0	Numeric	3	0	N
25	EVENTNOLC1	Character	2		N
26	LC1	Character	1		N
27	AVERAGELC1	Numeric	3	0	N
28	PEAKLC1	Numeric	3	0	N
29	LASTLC1	Numeric	3	0	N
30	EVENTNOLC2	Character	2		N
31	LC2	Character	1		N
32	AVERAGELC2	Numeric	3	0	N
33	PEAKLC2	Numeric	3	0	N
34	LASTLC2	Numeric	3	0	N
35	EVENTNOLC3	Character	2		N
36	LC3	Character	1		N
37	AVERAGELC3	Numeric	3	0	N
38	PEAKLC3	Numeric	3	0	N
39	LASTLC3	Numeric	3	0	N

- **Link Statistics**

XPLEX11.TXT

Num	Field Name	Field Type	Width	Dec	Index
1	DATE	Date	8		Y
2	HOUR	Character	2		Y
3	MINUTES	Character	2		Y
4	COUNT	Character	3		N
5	DEVICETYPE	Character	1		N
6	NODE	Character	3		Y
7	LINK	Character	3		Y
8	MINS	Numeric	5	0	N
9	EVENTNO23	Character	2		N
10	TRANS23	Numeric	5	0	N
11	RECVE23	Numeric	5	0	N
12	EVENTNO24	Character	2		N
13	TRANS24	Numeric	5	0	N
14	RECVE24	Numeric	5	0	N
15	EVENTNO25	Character	2		N
16	TRANS25	Numeric	5	0	N
17	RECVE25	Numeric	5	0	N
18	EVENTNO26	Character	2		N
19	TRANS26	Numeric	5	0	N
20	RECVE26	Numeric	5	0	N
21	EVENTNO27	Character	2		N
22	AVERAGE27	Numeric	3	0	N
23	PEAK27	Numeric	3	0	N
24	LAST27	Numeric	3	0	N
25	EVENTNO28	Character	2		N
26	AVERAGE28	Numeric	3	0	N
27	PEAK28	Numeric	3	0	N
28	LAST28	Numeric	3	0	N
29	EVENTNO29	Character	2		N
30	AVERAGE29	Numeric	3	0	N
31	PEAK29	Numeric	3	0	N
32	LAST29	Numeric	3	0	N

XPLEX12.TXT

Num	Field Name	Field Type	Width	Dec	Index
1	DATE	Date	8		Y
2	HOUR	Character	2		Y
3	MINUTES	Character	2		Y
4	COUNT	Character	3		N
5	DEVICETYPE	Character	1		N
6	EVENTNUMB	Character	2		N
7	ORIGNODE	Character	3		Y
8	ORIGLINK	Character	3		Y
9	ORIGCHAN	Character	3		Y
10	NODE	Character	3		Y
11	LINK	Character	3		Y
12	CHANNEL	Character	3		Y

XPLEX13.TXT

Num	Field Name	Field Type	Width	Dec	Index
1	DATE	Date	8		Y
2	HOUR	Character	2		Y
3	MINUTES	Character	2		Y
4	COUNT	Character	3		N
5	DEVICETYPE	Character	1		N
6	EVENTNUMB	Character	2		N
7	NODE	Character	3		Y
8	LC	Character	1		Y
9	MINS	Numeric	5	0	N
10	AVERAGE	Numeric	3	0	N
11	PEAK	Numeric	3	0	N
12	LAST	Numeric	3	0	N

5.1.3 NETWORK CONFIGURATION FILES.

NODE.DBF

Num	Field Name	Field Type	Width	Dec	Index
1	NODE	Character	3		Y
2	NNAME	Character	4		N
3	NODENAME	Character	22		N

LINK.DBF

Num	Field Name	Field Type	Width	Dec	Index
1	NODE	Character	3		Y
2	LINK	Character	3		Y
3	LNAME	Character	4		N
4	LINKNAME	Character	22		N

CHANNEL.DBF

Num	Field Name	Field Type	Width	Dec	Index
1	NODE	Character	3		Y
2	LINK	Character	3		Y
3	CHANNEL	Character	3		Y
4	CNAME	Character	4		N
5	CHANNAME	Character	22		N

MEMVARS.DBF

Num	Field Name	Field Type	Width	Dec	Index
1	FROMDATEMV	Date	8		N
2	TODATEMV	Date	8		N
3	FROMHOURMV	Character	2		N
4	TOHOURMV	Character	2		N
5	FROMMINSMV	Character	2		N
6	TOMINSMV	Character	2		N
7	FROMNODEMV	Character	3		N
8	TONODEMV	Character	3		N
9	FROMLINKMV	Character	3		N
10	TOLINKMV	Character	3		N
11	FROMCHANMV	Character	3		N
12	TOCHANMV	Character	3		N

DEVICE.DBF

Num	Field Name	Field Type	Width	Dec	Index
1	DEVICETYPE	Character	1		Y
2	DEVICECODE	Character	3		N
3	TYPEOFDEV	Character	10		N

5.2 Database Query Structures

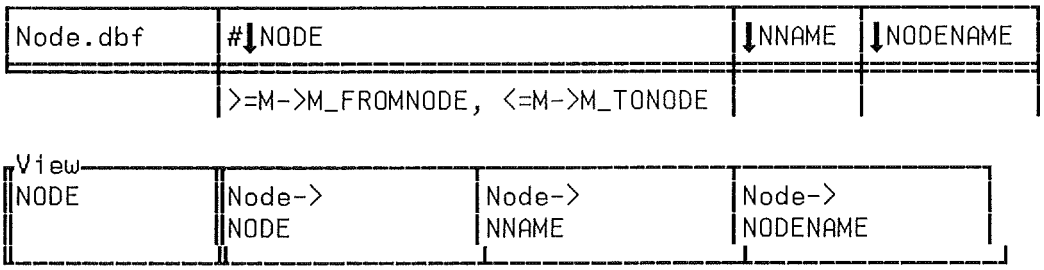
Dbase IV uses Query By Example as its query language. It does support SQL but all of the queries in the application use the Query By Example procedures.

The queries are shown in their corresponding sections, these are:

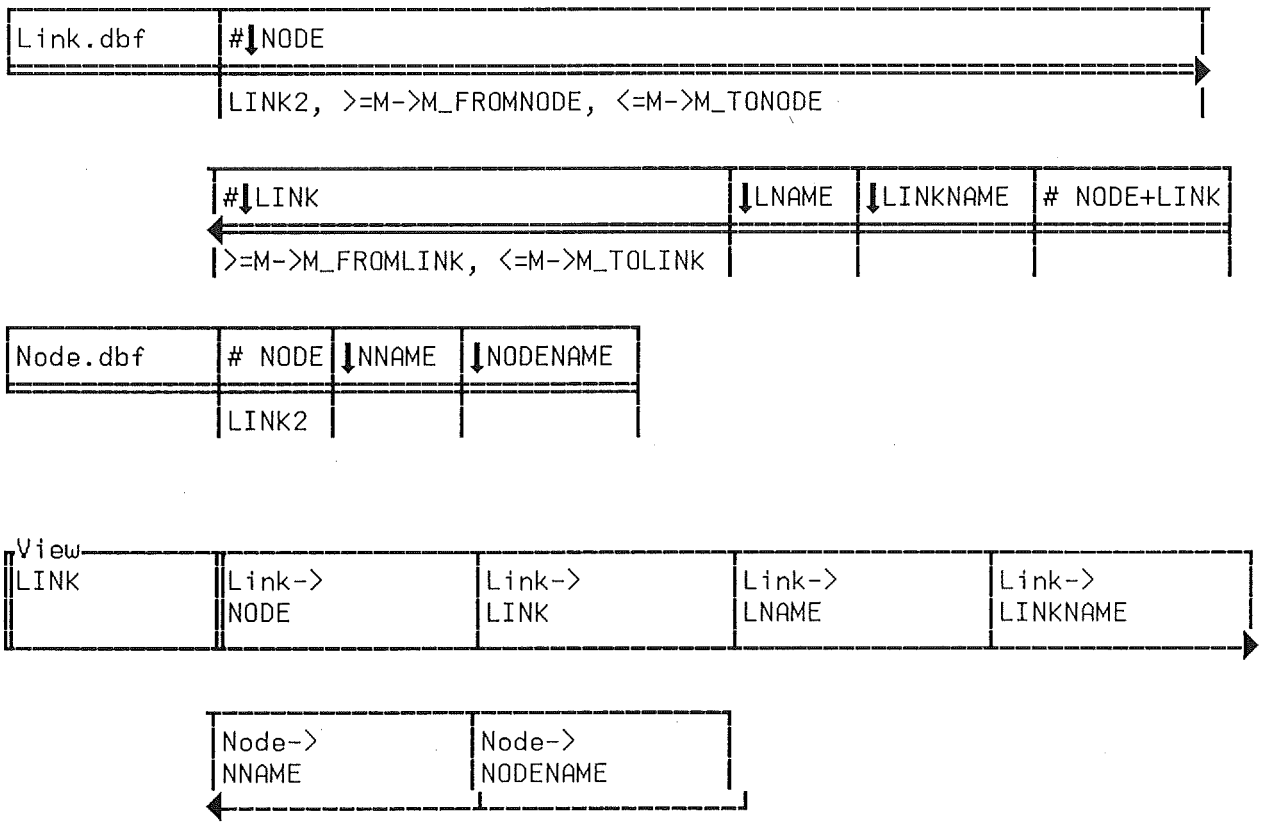
3.2.1 Configuration
3.2.2 Statistics

5.2.1 Configuration.

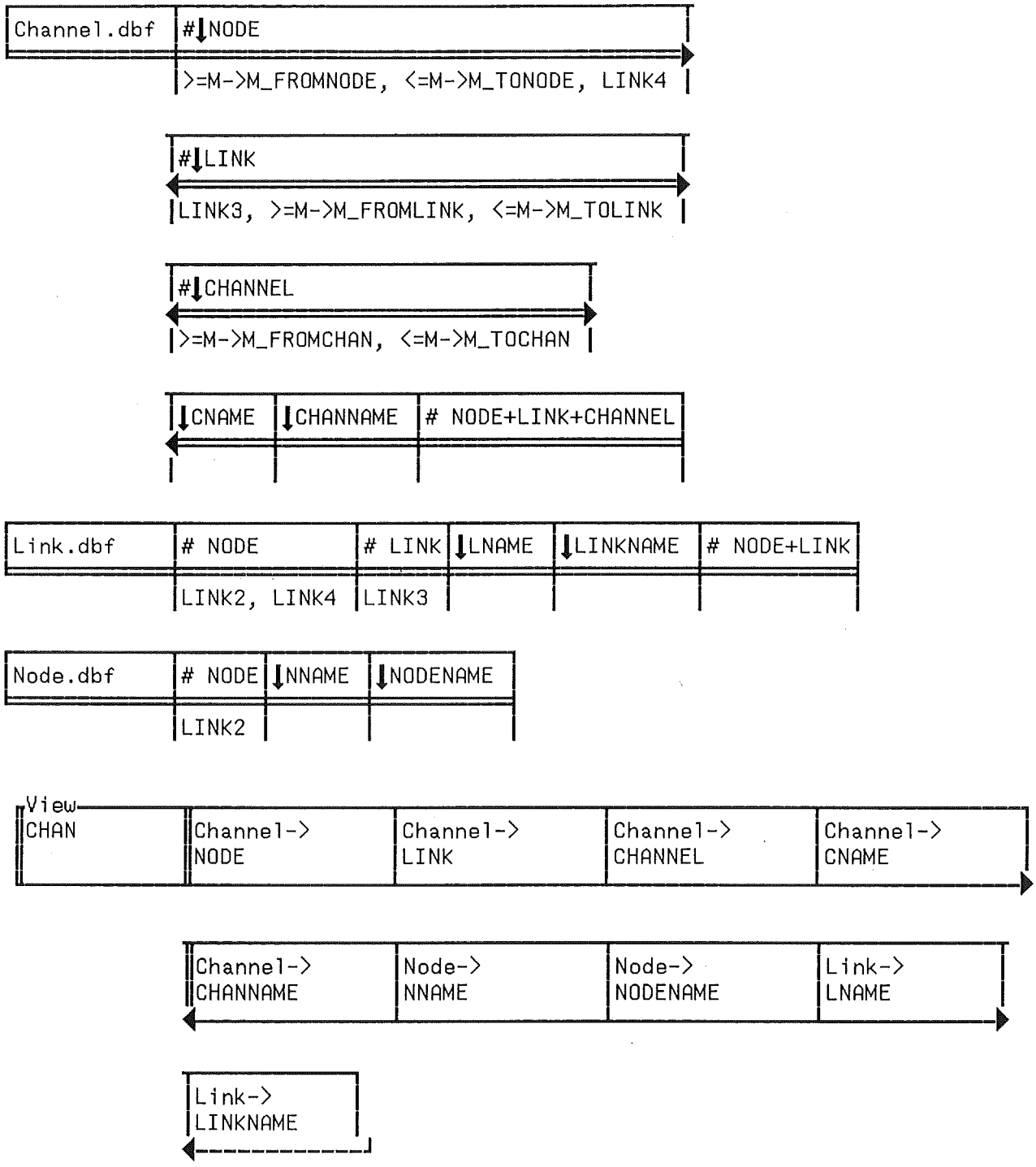
- **NODE.QBE**



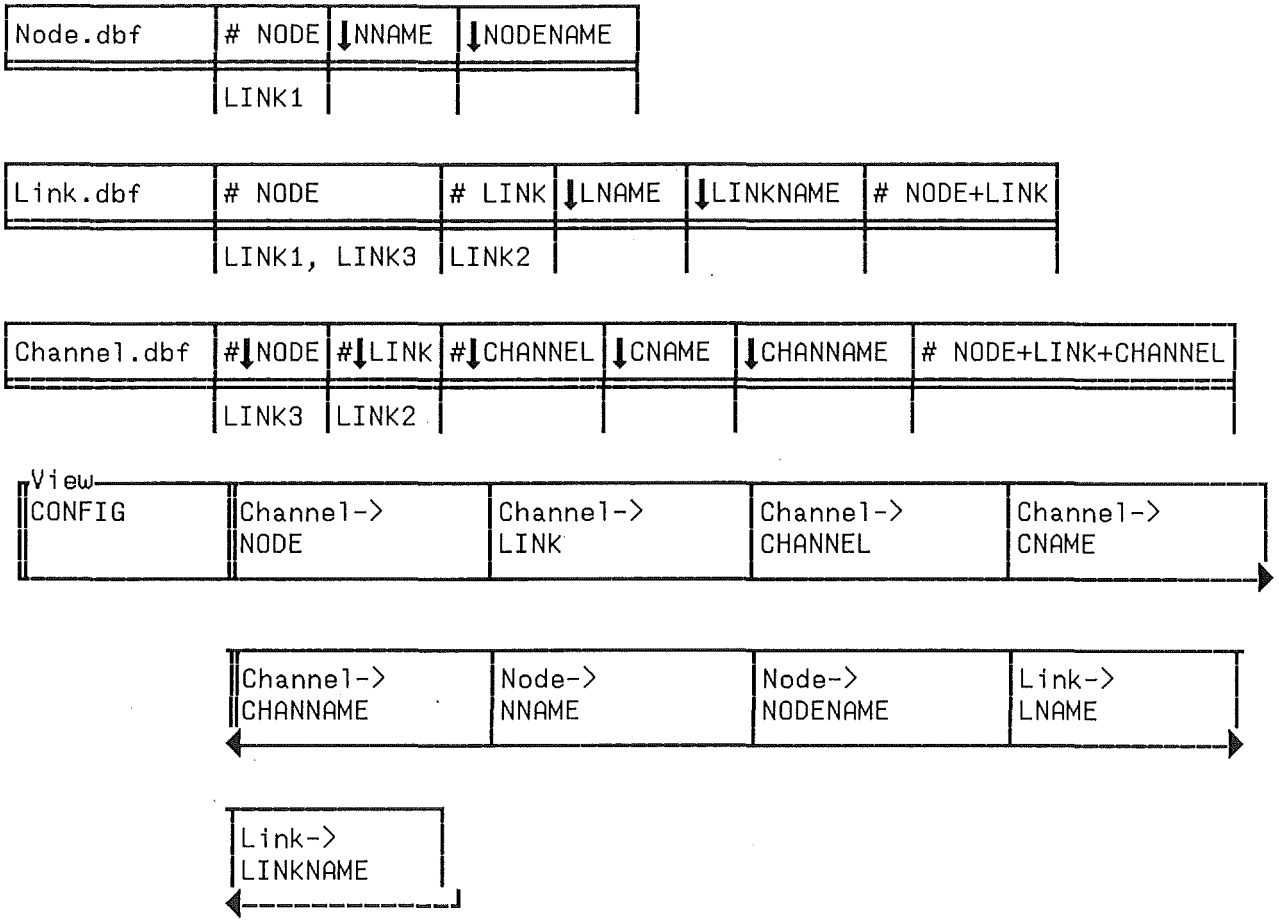
- **LINK.QBE**



· CHAN.QBE

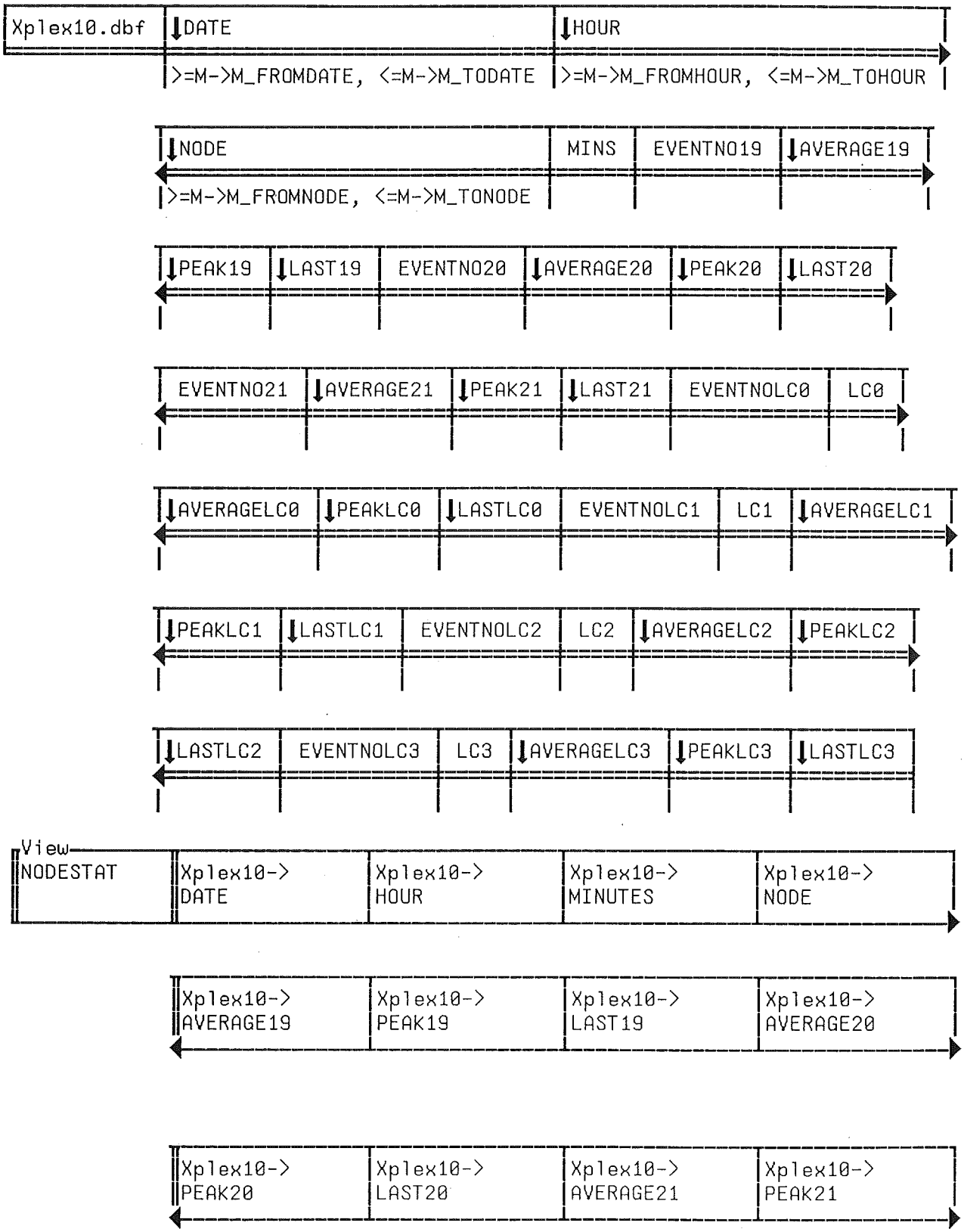

Channel.dbf
#↓NODE
>=M->M_FROMNODE, <=M->M_TONODE, LINK4
#↓LINK
LINK3, >=M->M_FROMLINK, <=M->M_TOLINK
#↓CHANNEL
>=M->M_FROMCHAN, <=M->M_TOCHAN
↓CNAME
↓CHANNAME
NODE+LINK+CHANNEL
Link.dbf
NODE
LINK
↓LNAME
↓LINKNAME
NODE+LINK
LINK2, LINK4
LINK3
Node.dbf
NODE
↓NNAME
↓NODENAME
LINK2
View
CHAN
Channel->
NODE
Channel->
LINK
Channel->
CHANNEL
Channel->
CNAME
Channel->
CHANNAME
Node->
NNAME
Node->
NODENAME
Link->
LNAME
Link->
LINKNAME

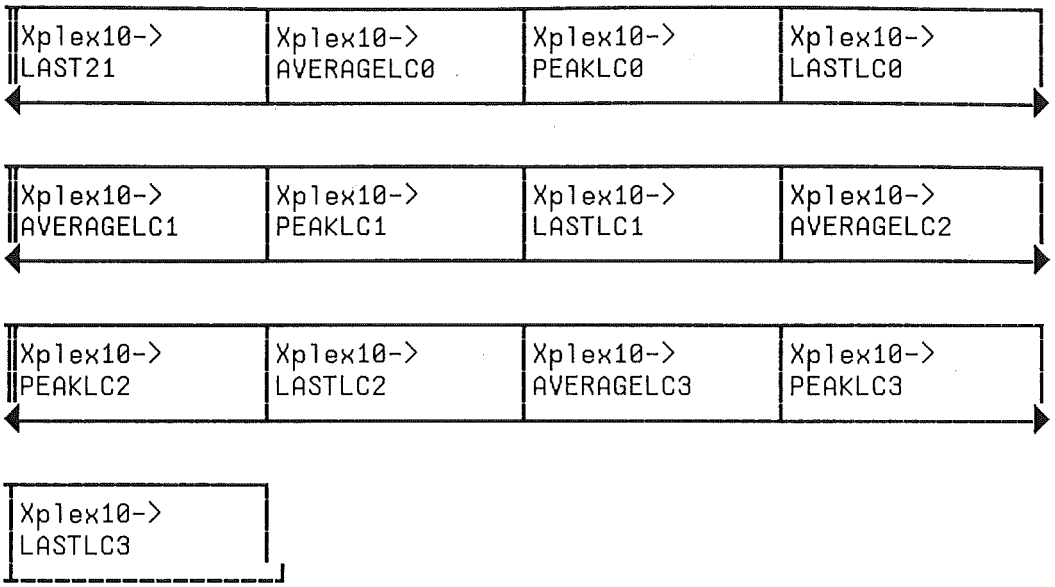
· **CONFIG.QBE**


Node.dbf
NODE
↓NNAME
↓NODENAME
LINK1
Link.dbf
NODE
LINK
↓LNAME
↓LINKNAME
NODE+LINK
LINK1, LINK3
LINK2
Channel.dbf
#↓NODE
#↓LINK
#↓CHANNEL
↓CNAME
↓CHANNAME
NODE+LINK+CHANNEL
LINK3
LINK2
View
CONFIG
Channel->
NODE
Channel->
LINK
Channel->
CHANNEL
Channel->
CNAME
Channel->
CHANNAME
Node->
NNAME
Node->
NODENAME
Link->
LNAME
Link->
LINKNAME

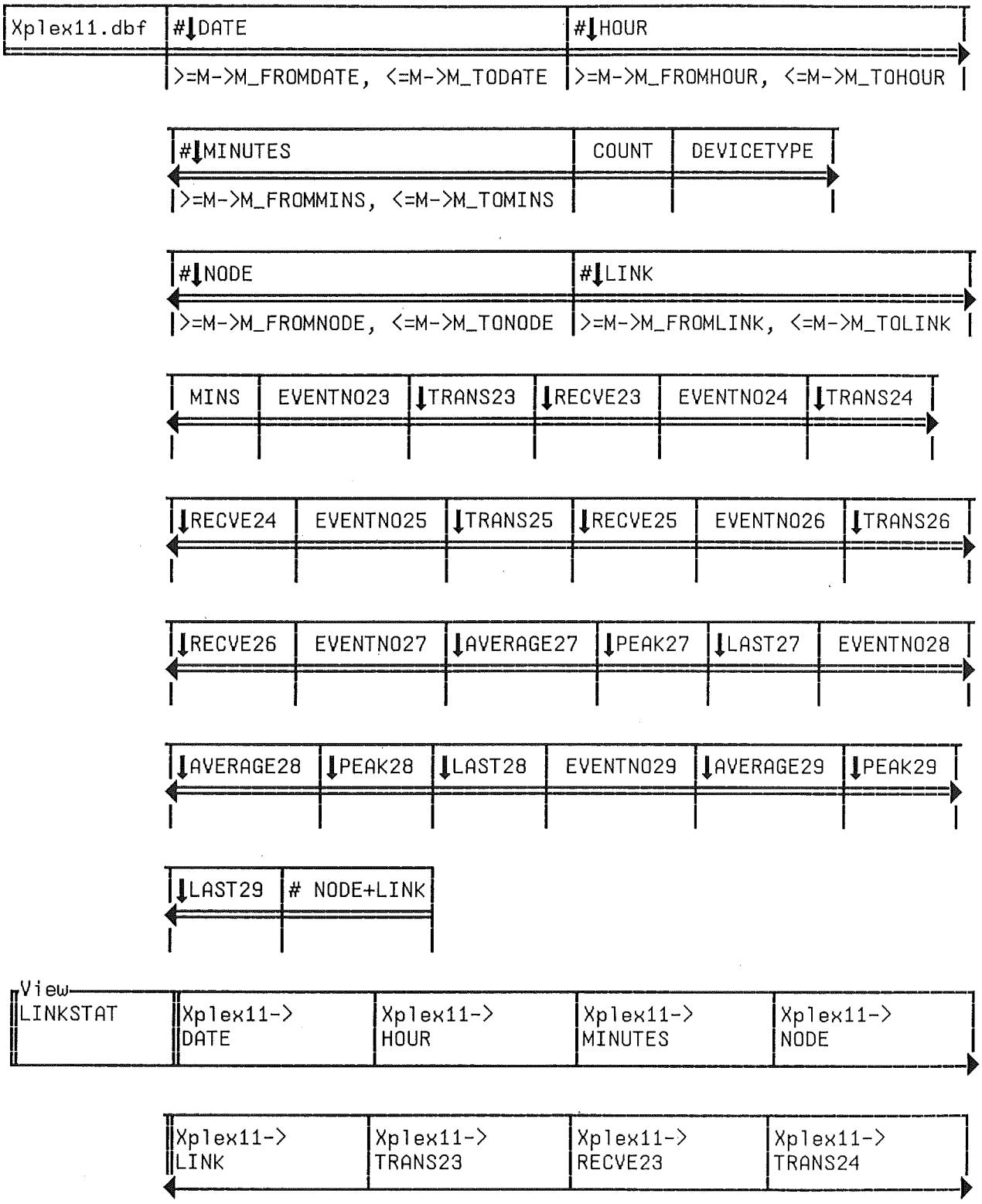
5.2.2 Statistics.

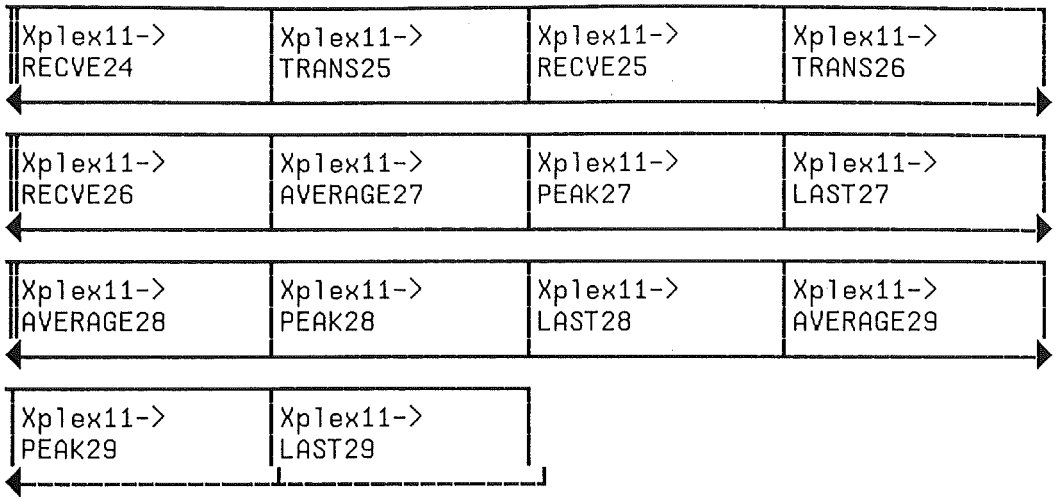
- **Xplexer Statistics**
  - **NODESTAT.QBE**



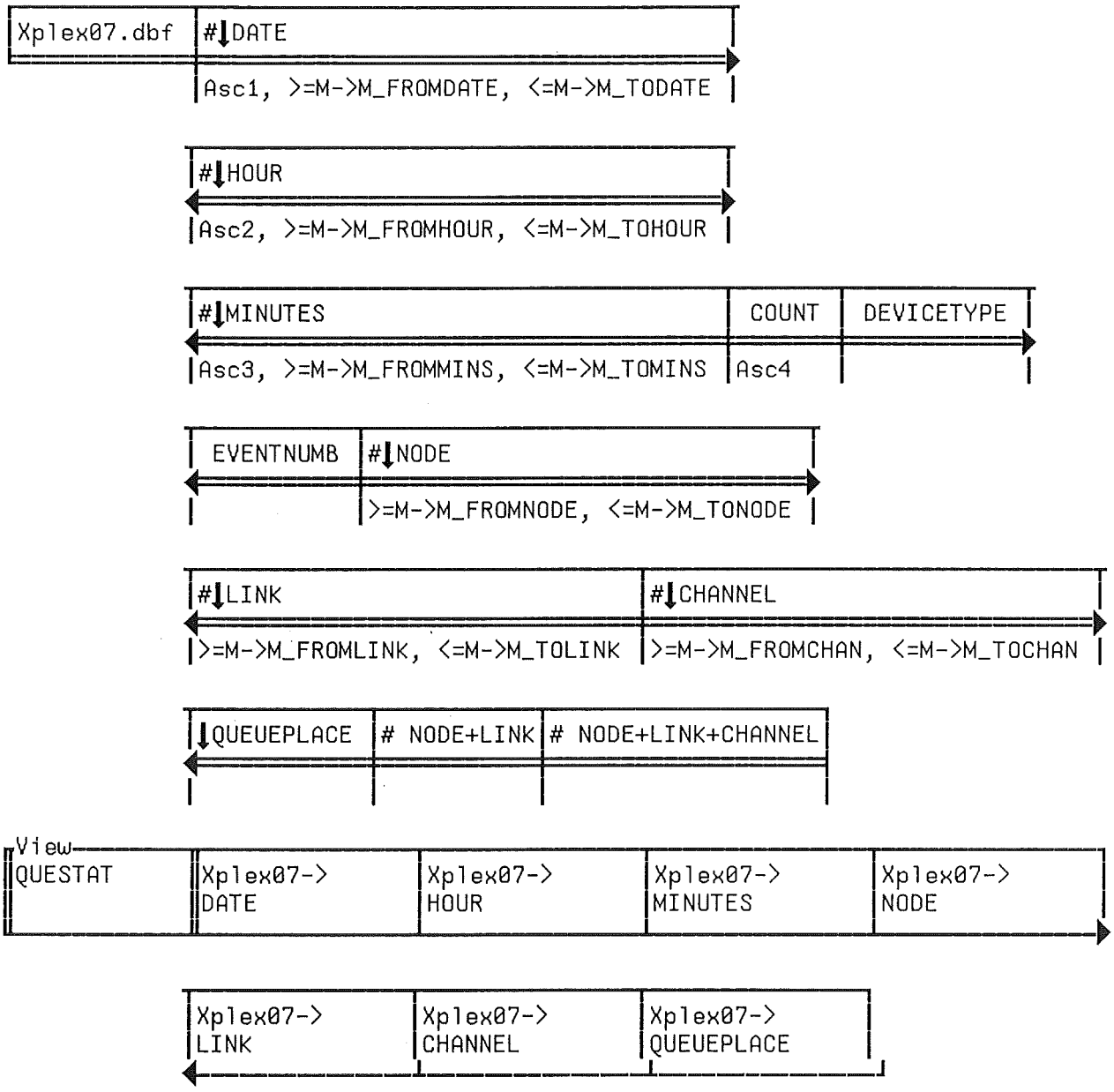

Xplex10->
LAST21
Xplex10->
AVERAGELC0
Xplex10->
PEAKLC0
Xplex10->
LASTLC0
Xplex10->
AVERAGELC1
Xplex10->
PEAKLC1
Xplex10->
LASTLC1
Xplex10->
AVERAGELC2
Xplex10->
PEAKLC2
Xplex10->
LASTLC2
Xplex10->
AVERAGELC3
Xplex10->
PEAKLC3
Xplex10->
LASTLC3

- **LINKSTAT.QBE**

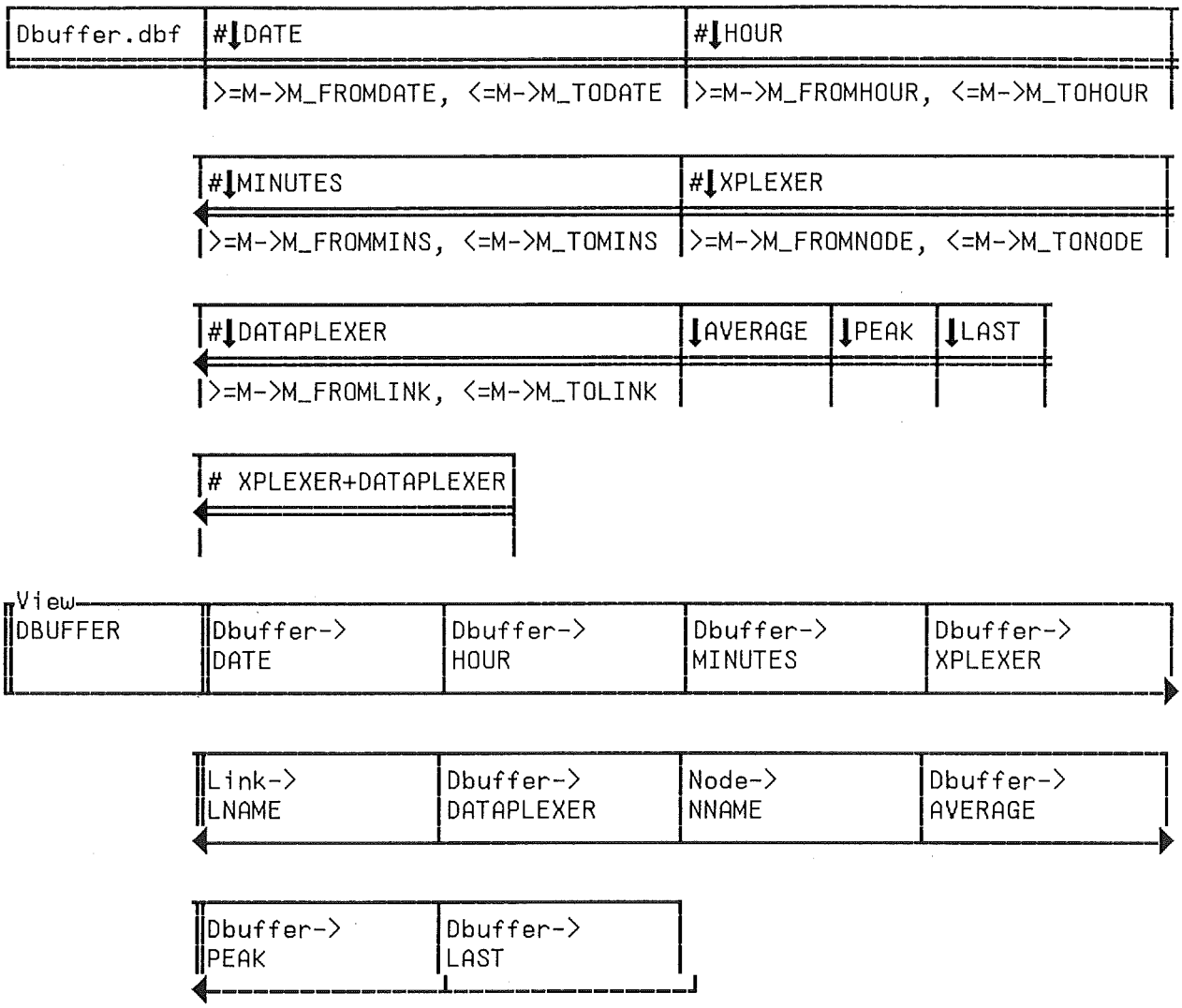



Xplex11->
RECVE24
Xplex11->
TRANS25
Xplex11->
RECVE25
Xplex11->
TRANS26
Xplex11->
RECVE26
Xplex11->
AVERAGE27
Xplex11->
PEAK27
Xplex11->
LAST27
Xplex11->
AVERAGE28
Xplex11->
PEAK28
Xplex11->
LAST28
Xplex11->
AVERAGE29
Xplex11->
PEAK29
Xplex11->
LAST29

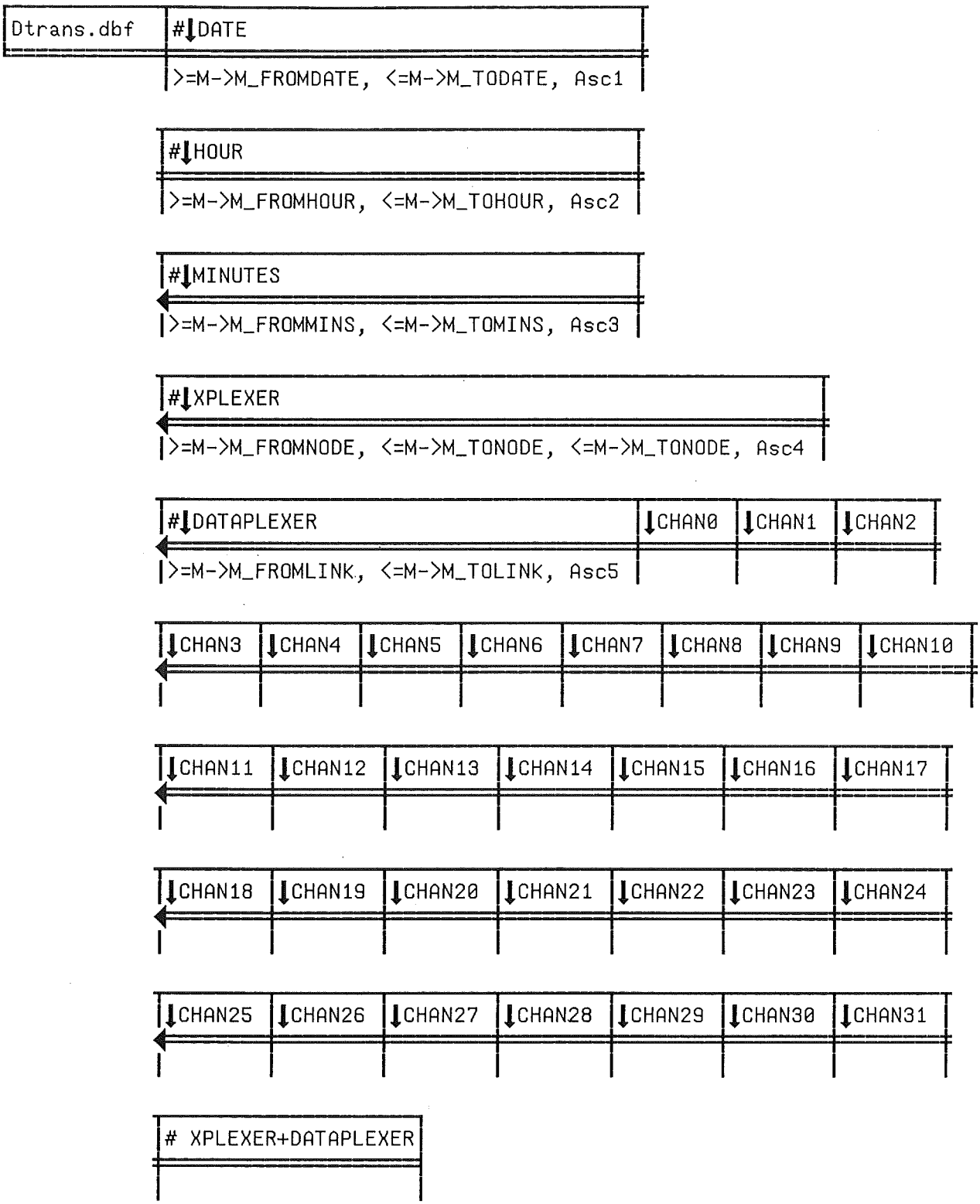
- **QUESTAT.QBE**

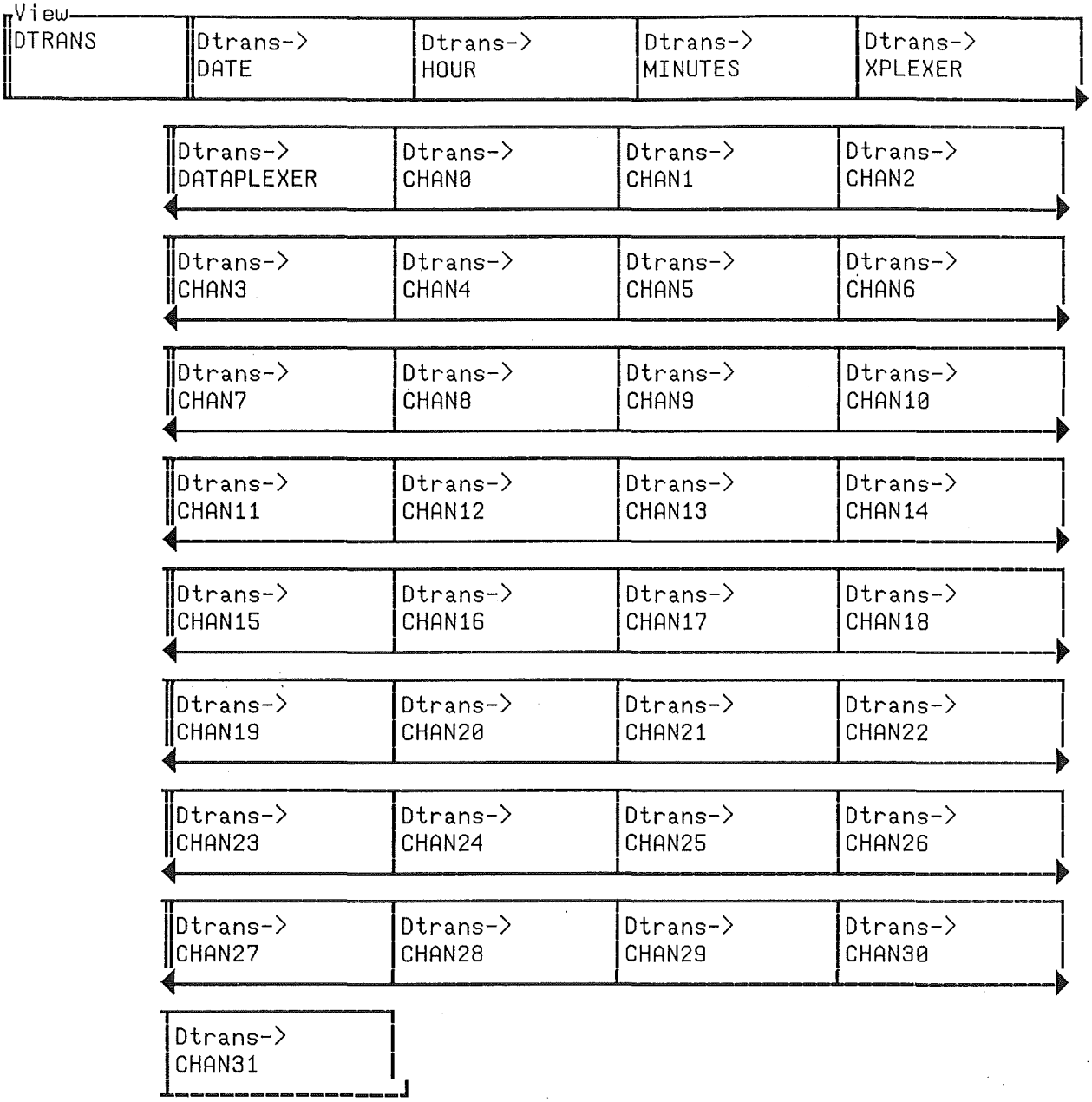

Xplex07.dbf
#↓DATE
Asc1, >=M->M_FROMDATE, <=M->M_TODATE
#↓HOUR
Asc2, >=M->M_FROMHOUR, <=M->M_TOHOUR
#↓MINUTES
COUNT
DEVICETYPE
Asc3, >=M->M_FROMMINS, <=M->M_TOMINS
Asc4
EVENTNUMB
#↓NODE
>=M->M_FROMNODE, <=M->M_TONODE
#↓LINK
#↓CHANNEL
>=M->M_FROMLINK, <=M->M_TOLINK
>=M->M_FROMCHAN, <=M->M_TOCHAN
↓QUEUEPLACE
NODE+LINK
NODE+LINK+CHANNEL
View
QUESTAT
Xplex07->
DATE
Xplex07->
HOUR
Xplex07->
MINUTES
Xplex07->
NODE
Xplex07->
LINK
Xplex07->
CHANNEL
Xplex07->
QUEUEPLACE

- **Dataplexer Statistics**
  - **DBUFFER.QBE**

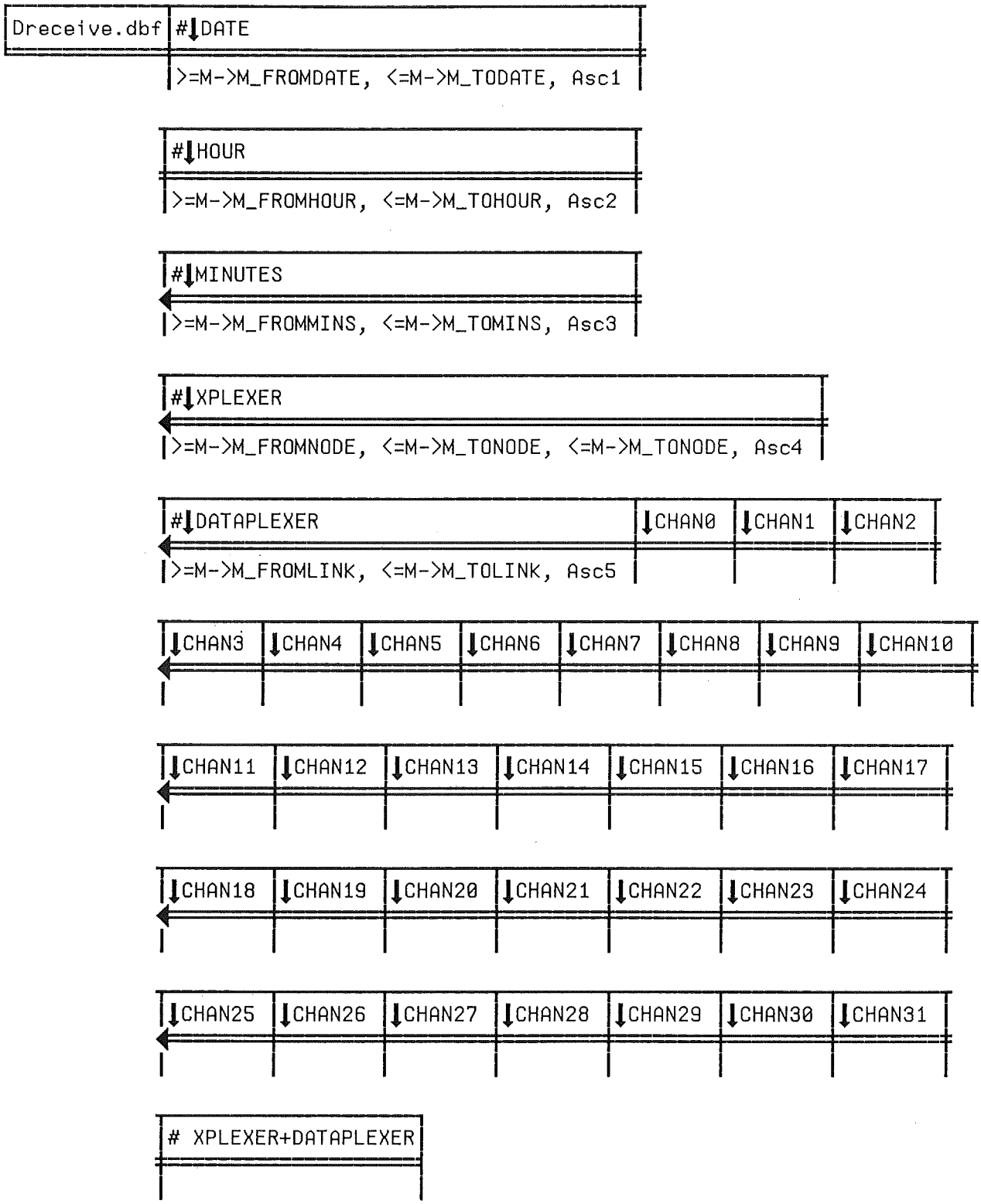


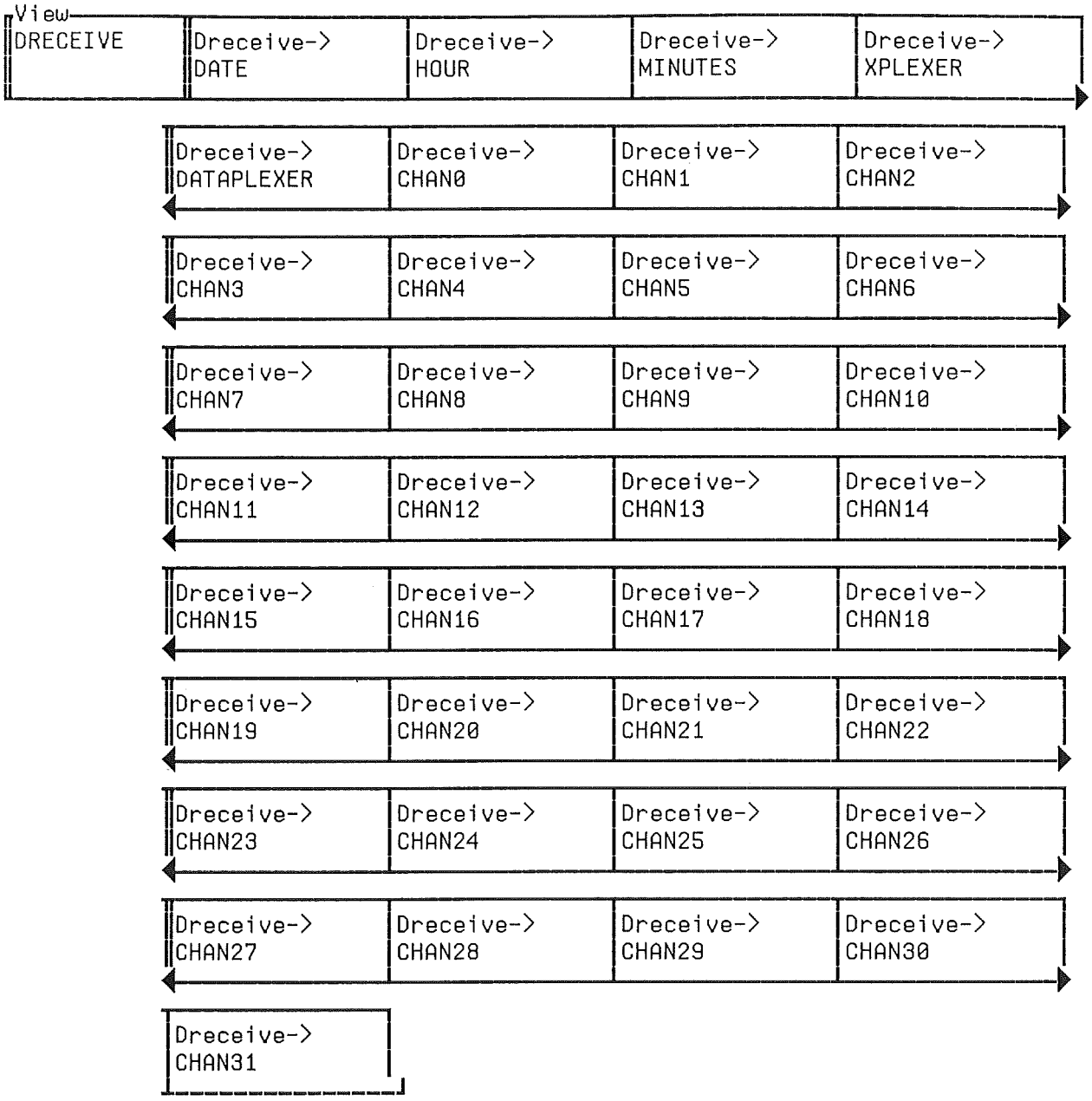
· DTRANS.QBE


Dtrans.dbf
#↓DATE
>=M->M_FROMDATE, <=M->M_TODATE, Asc1
#↓HOUR
>=M->M_FROMHOUR, <=M->M_TOHOUR, Asc2
#↓MINUTES
>=M->M_FROMMINS, <=M->M_TOMINS, Asc3
#↓XPLEXER
>=M->M_FROMNODE, <=M->M_TONODE, <=M->M_TONODE, Asc4
#↓DATAPLEXER
↓CHAN0
↓CHAN1
↓CHAN2
>=M->M_FROMLINK, <=M->M_TOLINK, Asc5
↓CHAN3
↓CHAN4
↓CHAN5
↓CHAN6
↓CHAN7
↓CHAN8
↓CHAN9
↓CHAN10
↓CHAN11
↓CHAN12
↓CHAN13
↓CHAN14
↓CHAN15
↓CHAN16
↓CHAN17
↓CHAN18
↓CHAN19
↓CHAN20
↓CHAN21
↓CHAN22
↓CHAN23
↓CHAN24
↓CHAN25
↓CHAN26
↓CHAN27
↓CHAN28
↓CHAN29
↓CHAN30
↓CHAN31
XPLEXER+DATAPLEXER


View
DTRANS
Dtrans->
DATE
Dtrans->
HOUR
Dtrans->
MINUTES
Dtrans->
XPLEXER
Dtrans->
DATAPLEXER
Dtrans->
CHAN0
Dtrans->
CHAN1
Dtrans->
CHAN2
Dtrans->
CHAN3
Dtrans->
CHAN4
Dtrans->
CHAN5
Dtrans->
CHAN6
Dtrans->
CHAN7
Dtrans->
CHAN8
Dtrans->
CHAN9
Dtrans->
CHAN10
Dtrans->
CHAN11
Dtrans->
CHAN12
Dtrans->
CHAN13
Dtrans->
CHAN14
Dtrans->
CHAN15
Dtrans->
CHAN16
Dtrans->
CHAN17
Dtrans->
CHAN18
Dtrans->
CHAN19
Dtrans->
CHAN20
Dtrans->
CHAN21
Dtrans->
CHAN22
Dtrans->
CHAN23
Dtrans->
CHAN24
Dtrans->
CHAN25
Dtrans->
CHAN26
Dtrans->
CHAN27
Dtrans->
CHAN28
Dtrans->
CHAN29
Dtrans->
CHAN30
Dtrans->
CHAN31

- **DRECEIVE.QBE**




View
DRECEIVE
Dreceive->
DATE
Dreceive->
HOUR
Dreceive->
MINUTES
Dreceive->
XPLEXER
Dreceive->
DATAPLEXER
Dreceive->
CHAN0
Dreceive->
CHAN1
Dreceive->
CHAN2
Dreceive->
CHAN3
Dreceive->
CHAN4
Dreceive->
CHAN5
Dreceive->
CHAN6
Dreceive->
CHAN7
Dreceive->
CHAN8
Dreceive->
CHAN9
Dreceive->
CHAN10
Dreceive->
CHAN11
Dreceive->
CHAN12
Dreceive->
CHAN13
Dreceive->
CHAN14
Dreceive->
CHAN15
Dreceive->
CHAN16
Dreceive->
CHAN17
Dreceive->
CHAN18
Dreceive->
CHAN19
Dreceive->
CHAN20
Dreceive->
CHAN21
Dreceive->
CHAN22
Dreceive->
CHAN23
Dreceive->
CHAN24
Dreceive->
CHAN25
Dreceive->
CHAN26
Dreceive->
CHAN27
Dreceive->
CHAN28
Dreceive->
CHAN29
Dreceive->
CHAN30
Dreceive->
CHAN31

```
/*                                                                    *\
*  TXTTOTMP.EXE - The contents of the text file are copied to a temp file          *
*                 *.TMP. Then the contents of the file specified is removed.  *
*                  The file is locked while this is done so there is no conflict    *
*                  over access to it. Once the file has been locked, the size of *
\*                 the file is changed to 0, and the lock is released.       */


# include <h:\users\lbw\c600\include\stdio.h>
# include <h:\users\lbw\c600\include\io.h>
# include <h:\users\lbw\c600\include\sys\locking.h>
# include <h:\users\lbw\c600\include\fcntl.h>
# include <h:\users\lbw\c600\include\process.h>
# include <h:\users\lbw\c600\include\share.h>

# define FALSE 0
# define S_IWRITE          0000200              /* write permission, owner */
# define S_IREAD   0000400              /* read permission, owner */

int handle;
long length;
char ch;

main(int argc, char *argv[])
{
while ((handleTXT = sopen(argv[1], O_CREAT | O_RDWR, SH_DENYNO, S_IREAD |
S_IWRITE)) == FALSE);
while ((handleTMP = sopen(argv[2], O_CREAT | O_WRONLY, SH_DENYNO, S_IWRITE)) ==
FALSE);
length = filelength(handleTXT);
while(locking(handleTXT, LK_NBRLCK, length) != -1);
lseek(handleTXT, 0L, SEEK_SET);
while((read(handleTXT, &ch, 1)) == 1)
    write(handleTMP, &ch, 1);
close(handleTMP);
chsize(handleTXT, 0);
while(locking(handleTXT, LK_UNLCK, length) != -1);
close(handleTXT);
}
```

```
/*                                                              *\
*   DELFILES - The contents of the file specified as the argument is removed.*
*              The file is locked while this is done so there is no conflict *
*              over access to it. Once the file has been locked, the size of *
\*             the file is changed to 0, and the lock is released.          */


# include <h:\users\lbw\c600\include\stdio.h>
# include <h:\users\lbw\c600\include\io.h>
# include <h:\users\lbw\c600\include\sys\locking.h>
# include <h:\users\lbw\c600\include\fcntl.h>
# include <h:\users\lbw\c600\include\process.h>
# include <h:\users\lbw\c600\include\share.h>

# define FALSE 0
# define S_IWRITE          0000200              /* write permission, owner */
# define S_IREAD   0000400              /* read permission, owner */

int handle;
long length;
char ch;

main(int argc, char *argv[])
{
while ((handleTXT = sopen(argv[1], O_CREAT | O_RDWR, SH_DENYNO, S_IREAD |
S_IWRITE)) == FALSE);
while ((handleTMP = sopen(argv[2], O_CREAT | O_WRONLY, SH_DENYNO, S_IWRITE)) ==
FALSE);
length = filelength(handleTXT);
while(locking(handleTXT, LK_NBRLCK, length) != -1);
lseek(handleTXT, 0L, SEEK_SET);
while((read(handleTXT, &ch, 1)) == 1)
    write(handleTMP, &ch, 1);
close(handleTMP);
chsize(handleTXT, 0);
while(locking(handleTXT, LK_UNLCK, length) != -1);
close(handleTXT);
}
```